PARAMETRIC STRUCTURAL MODELING

User Manual for Version 0.9.007

written by Clemens Preisinger
contributions by Justin Diles and Robert Vierlinger

September 7, 2011

Contents

1 Introduction

Karamba is a Finite Element plug-in for the Rhino plug-in Grasshopper. It lets you interactively calculate the response of three dimensional beam structures under the action of external loads.

Karamba is fully embedded in the parametric environment of Grasshopper. This makes it easy to combine parameterized geometric models, Karamba and optimization algorithms like Galapagos.

2 Disclaimer

Although being tested thoroughly Karamba probably contains errors – therefore no guarantee can be given that Karamba computes correct results. Use of Karamba is entirely at your own risk. Please read the licence agreement that comes with Karamba in case of further questions.

3 Installation

These are the prerequisites for installing Karamba:

- **Rhino 4.0**
  with Service Release 8 or 9
- **Grasshopper**
  Build 0.8.0051.

In case you do not possess Rhino 4 download a fully featured, free trial version of from `http://www.rhino3d.com/download.html`. Grasshopper which is free can be found at `http://www.grasshopper3d.com/`.

Other versions of Grasshopper than that mentioned above may cause problems. We will try to keep up with Grasshoppers future development by providing corresponding Karamba versions. Check the download area of the Karamba web-site for corresponding information.

Invoke "KarambaSetup.msi" and choose the folder where Grasshopper is installed when asked for an installation path. This is usually something like "C://Programs/Rhinoceros 4.0/Plug-ins/Grasshopper". Besides other things a folder named "Karamba" will be created there, which contains the license agreement, a readme-file, pre-fabricated cross section and material tables and the configuration file "karamba.ini". The config file can be edited with any text editor. It contains key - value pairs and is pretty self descriptive.

If all goes well you will notice upon starting Grasshopper that there is a new category called Karamba on the component panel. It consists of eight subsections (see figure 1). In case you do not see any icons select "Draw All Components" in Grasshoppers "View"-menu. If you consistently get an "fem.karambaPINVOKE" exception see section 7.2 for how to solve that issue.

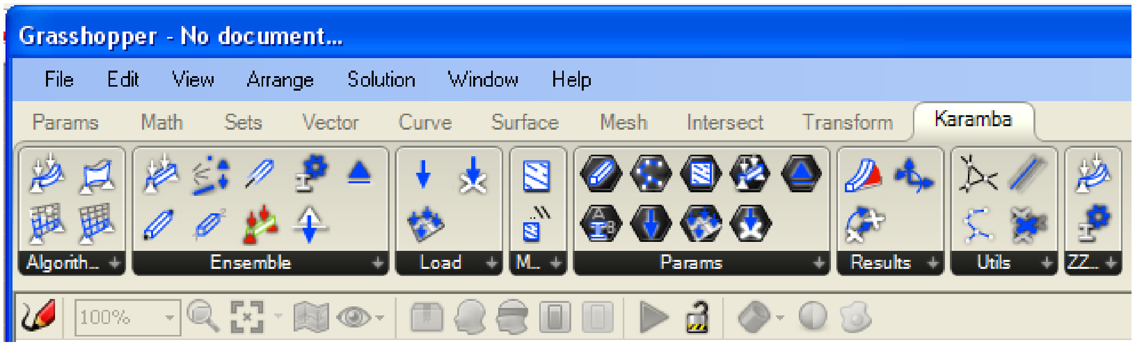


Figure 1: Category "Karamba" on the component panel

These are the subsections which show up in the Karamba category:

- "Algorithms": components for the calculation of statical models
- "Cross Section": contains components to create and select cross sections for beams.
- "Ensemble": lets you create models
- "Loads": components for applying external forces
- "Materials": components for the definition of material properties
- "Params": classes of objects that make up the statical model
- "Results": for the retrieval of calculation results
- "Utils": contains some extra geometric functionality that is not directly linked to creating a model but makes certain things easy.
- "ZZZ-Depricated": holds components which changed from an earlier version of Karamba to the current. They will be removed in some future version. In order to make definitions based on old versions work correctly they remain part of Karamba for the time being.

The colors of Karambas icons have a special meaning: black or white designates the entity or entities on which a component acts. Products of components get referenced by a blue symbol.

This guide assumes that you have some basic knowledge of Rhino and Grasshopper. In case you need introductory material regarding Grasshopper it is probably a good idea to download the "Grasshopper Primer" from the Grasshopper web-site.

4 Quick start

Creating a statical model in Karamba consists of six basic steps[1] – see fig. 2:

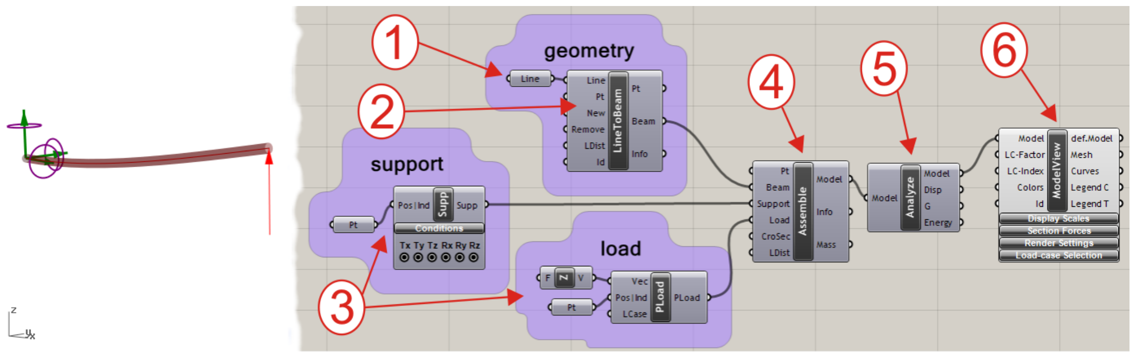


Figure 2: Basic example of a statical model in Karamba

1. **Create** wire-frame or point geometry for the structural model with Rhino or GH.

2. **Convert** wire-frame or point geometry to Karamba beams.

3. **Define** which points are supports and which receive loads. Optional: Define custom beam cross sections and multiple load cases.

4. **Assemble** the Karamba structural model with points, beams, supports and loads.

5. **Analyze** the Karamba structural model.

6. **View** the analyzed model. Deflections can be scaled, stress, strain, etc. can be observed and multiple load cases can be viewed together or

[1]This step-by-step procedure was devised and formulated by Justin Diles

separately. Mesh representations of beams can be refined to the desired resolution.

Karamba is intended to provide an intuitive approach to statical modeling. All its components come with extensive help-tags and there are lots of examples on the web-site which can be easily customized according to ones own needs. So casual users can do without reading further.

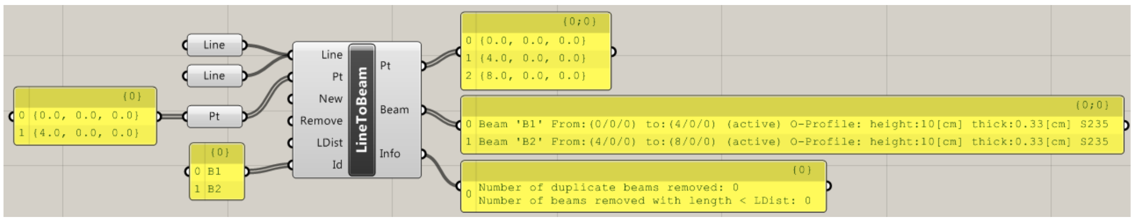
5 Usage

5.1 Define the model geometry

First thing to do when setting up a statical model is to define its geometry. In Rhino a given geometry consists of a collection of entities like points, lines and surfaces. For Karamba a given geometry consists of straight beams with physical properties (e.g. cross section, material). The next three subsections explain how to create them.

5.1.1 The LineToBeam-component

Figure 3 shows how Karambas LineToBeam-component takes two lines as input, finds out how they connect and outputs beams as well as a set of unique points which are their end-points. Points count as identical if their common distance is less than that given in "LDist" the default value being $0.005[m]$. The LineToBeam-component accepts only straight lines as geometric input. Therefore polylines and the like need to be exploded into segments first.



```
{0}
0 {0.0, 0.0, 0.0}
1 {4.0, 0.0, 0.0}

{0}
0 B1
1 B2

{0;0}
0 {0.0, 0.0, 0.0}
1 {4.0, 0.0, 0.0}
2 {8.0, 0.0, 0.0}

{0;0}
0 Beam 'B1' From:(0/0/0) to:(4/0/0) (active) O-Profile: height:10[cm] thick:0.33[cm] S235
1 Beam 'B2' From:(4/0/0) to:(8/0/0) (active) O-Profile: height:10[cm] thick:0.33[cm] S235

{0}
0 Number of duplicate beams removed: 0
  Number of beams removed with length < LDist: 0
```

Figure 3: The LineToBeam-component that turns two lines into beams

All coordinates are in meter. In order to be of immediate use the beams come with a number of default values (see string-output in figure 3): "active" means that it will be included in the static model. Teh default cross section is a circular profile of diameter $10[cm]$ with a wall-thickness of $0.33[cm]$. The default material is steel of grade "S235".

The first beam corresponds to the first item in the list of input lines and so on. The order in which points appear in the output node-list is random by default. However it is sometimes advantageous to identify certain points by their list index in order to put loads on them or to define supports. This can be achieved by feeding a list of coordinates into the "Points"-plug. They will be placed at the beginning of the output nodes-list. So in order that the end-points of the structure in figure 3 have index 0 and 1 it is necessary to input a list of points with coordinates (0/0/0) and (8/0/0).

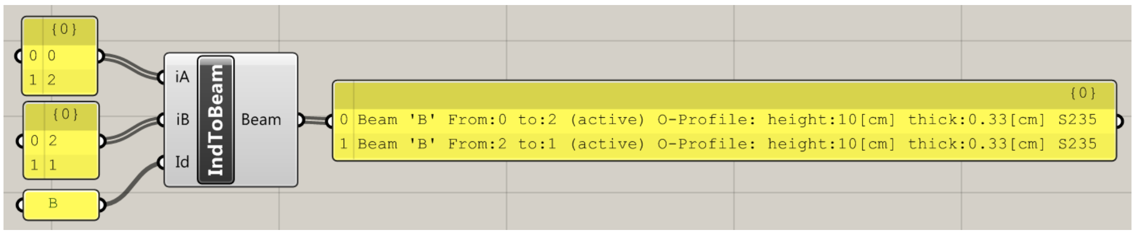
There are four more input plugs on the LineToBeam-component:

- New: If this plug has the value "False" only those lines will be added to the structure that start and end at one of the points given in the input points-list.
- Remove: If this option has the value "True" the LineToBeam-component checks for lines that lie on each other and merges such duplicates into one. This prevents an error that is hard to detect by visual inspection alone: Two lines on the same spot mean double member stiffness in the statical model.
- LDist: sets the limit distance for two points to be merged into one. Lines of length less than that value will be discarded.
- Id: takes a list of strings as identifiers for beams. If the number of items in this list is less than the number of beams then the last Id applies to the surplus beams. The default value is an empty string.

Beams that meet at a common point are connected rigidly in the statical model like they were welded together. The "Info" output-plug informs about the number of removed nodes and beams.

5.1.2 The IndToBeam component

Sometimes the initial geometry is already given as a set of points and two lists of node-indexes with one entry for each start- and end-point of beams respectively. In such a case it would be cumbersome to convert this information into geometric entities only for feeding it into the LineToBeam-component which reverses the previous step. The IndexToBeam-component (see figure



```
{0}
0 0
1 2
```

```
{0}
0 2
1 1
```

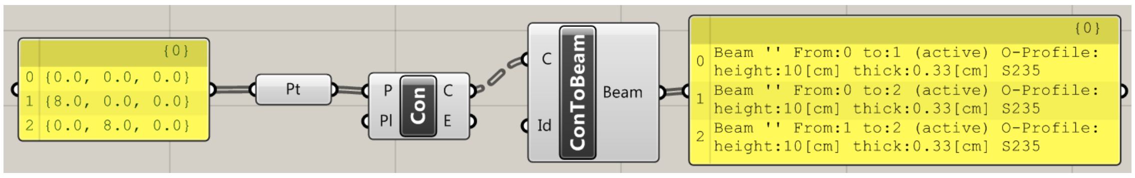
```
{0}
0 Beam 'B' From:0 to:2 (active) O-Profile: height:10[cm] thick:0.33[cm] S235
1 Beam 'B' From:2 to:1 (active) O-Profile: height:10[cm] thick:0.33[cm] S235
```

Figure 4: The IndexToBeam-component lets you directly define the connectivity information of beams

4) accepts a list of pairs of node-indexes and produces beams with default properties from it. This speeds up model generation considerably for there is no need to compare nodes for coincident coordinates.

5.1.3 The ConToBeam-component

In Grasshopper meshing algorithms result in topological connectivity diagrams. With the help of the ConToBeam-component these may be directly converted to beam-structures (see figure 5).



```
{0}
0 {0.0, 0.0, 0.0}
1 {8.0, 0.0, 0.0}
2 {0.0, 8.0, 0.0}
```

```
{0}
0 Beam '' From:0 to:1 (active) O-Profile:
  height:10[cm] thick:0.33[cm] S235
1 Beam '' From:0 to:2 (active) O-Profile:
  height:10[cm] thick:0.33[cm] S235
2 Beam '' From:1 to:2 (active) O-Profile:
  height:10[cm] thick:0.33[cm] S235
```

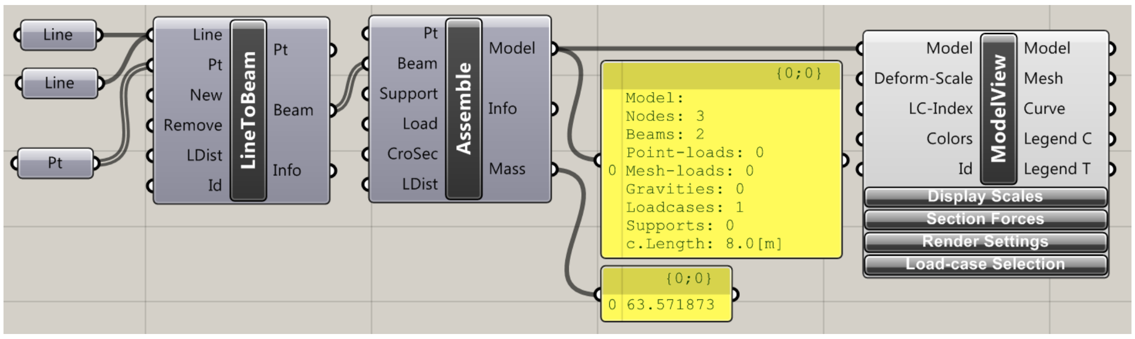
Figure 5: The ConToBeam-component turns connectivity diagrams into sets of beams

5.2 The Assemble-component

In order to calculate the behavior of a real world structure one needs to define its geometry, loads and supports. The component "Assemble" from the "Ensemble" subsection gathers all the necessary information and creates a statical model from it (see figure 6).

In case that some beams were defined by node indexes then these will refer to the list of points given at the "Pt" input-plug.

The output-plug "Mass" renders the mass of the structure in kilogram. When being plugged into a panel the model prints basic information about itself: number of nodes, beams, and so on. At the end of the list the characteristic length of the model is given which is calculated as the distance



```
{0;0}
0 Model:
  Nodes: 3
  Beams: 2
  Point-loads: 0
  Mesh-loads: 0
  Gravities: 0
  Loadcases: 1
  Supports: 0
  c.Length: 8.0[m]
```

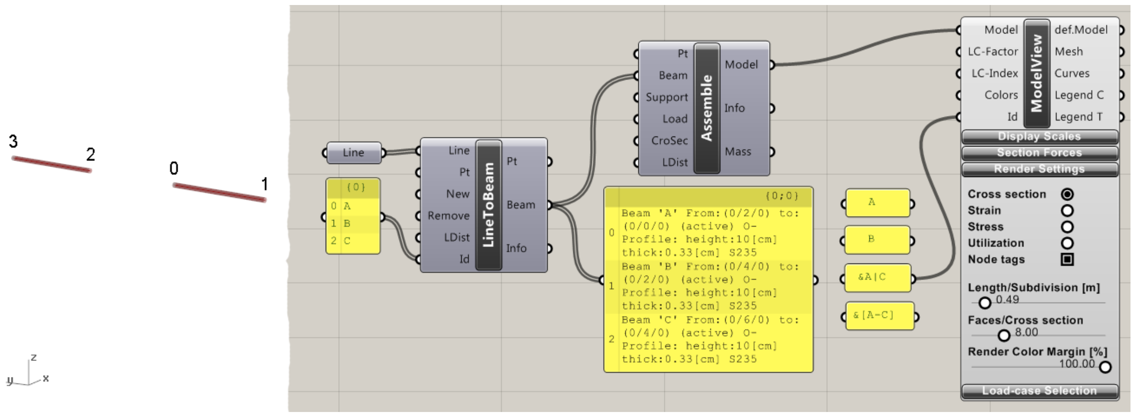
```
{0;0}
0 63.571873
```

Figure 6: The Assemble-component gathers data and creates a model from it.

between opposing corners of its bounding box.

5.3 The ModelView-component

The ModelView-component of the ”Ensemble” subsection can be used to check the current state of a statical model (see figure 6). When adding a ModelView to the definition it is a good idea to turn off the preview of all other components so that they do not interfere. Clicking on the black menu headings unfolds the ModelView and unveils additional widgets for tuning the model display. Each of these will be explained further below. The range and current value of the sliders may be set by double-clicking on the knob.



```
{0}
0 A
1 B
2 C
```

```
{0;0}
0 Beam 'A' From:(0/2/0) to:
  (0/0/0) (active) O-
  Profile: height:10[cm]
  thick:0.33[cm] S235
1 Beam 'B' From:(0/4/0) to:
  (0/2/0) (active) O-
  Profile: height:10[cm]
  thick:0.33[cm] S235
2 Beam 'C' From:(0/6/0) to:
  (0/4/0) (active) O-
  Profile: height:10[cm]
  thick:0.33[cm] S235
```

[✓ Cross section] Strain Stress Utilization [✓ Node tags]

Length/Subdivision [m]: 0.49
Faces/Cross section: 8.00
Render Color Margin [%]: 100.00

Figure 7: Partial view of a model.

The ModelView-component features five plugs on its left side:

- ”Model” expects the model to be displayed
- ”LC-Factor” can be used to scale individual load cases (see further

below).

- "LC-Index" lets one select the visible load-case (see example 'EModesWall.ghx'). The value in "LC-Index" will be added to the load-case selected in the drop-down-list of ModelView. If the resulting number is larger than the number of available load-cases the ModelView turns red. If the resulting value is smaller than 0 all load-cases are superimposed. The possibility of using a number-slider for selecting load-cases makes life easier in case that there are many of them.

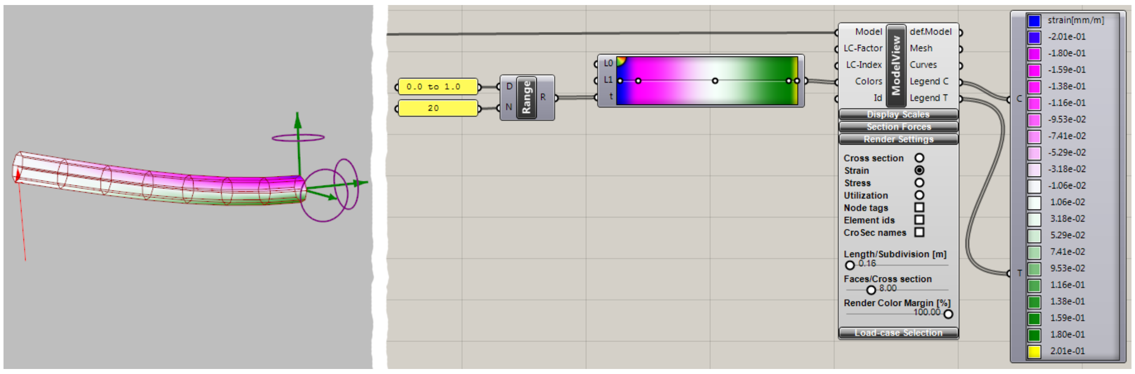
- "Colors": Color plots for stresses, strains etc. use a color spectrum from blue to white to red by default. One can customize the color range by handing over a list of RGB-values to the "Colors"-plug. There have to be at least three colors given. The first color is used for values below, the last color for values above the current number range. The remaining colors get evenly distributed over the number range. The Grasshopper component "Gradient" can be used to generate the list of colors (see fig. 8).

- "Id": This plug lets one select those parts of a model which shall be displayed. It expects a list of strings. The default value is an empty string which means that all of the model shall be visible. As one can see in fig. 7 it is possible to input regular expressions. These must start with the character "&" and adhere to the conventions of regular expressions as defined for C#. The identifier of each beam of the model is compared to each item of the given string list. In case a list entry matches the beam identifier the beam will be displayed. Fig. 7 contains four examples of "Id" lists: The first would limit visibility to beam "A", the second to beam "B". The third is a regular expression which matches beams "A" or "C". The fourth matches beams "A" to "C".

There are five output plugs on the ModelView-component:

- "Model" is essentially the model which was fed in on the left side. When there are results available from a statical calculation deflections are scaled and added to the node coordinates so that it contains the deformed geometry.

- From the "Mesh" output-plug you can get the mesh of the rendered model for further processing. In fact it is a list of meshes with each



Cross section ○
Strain ●
Stress ○
Utilization ○
Node tags ☐
Element ids ☐
CroSec names ☐
Length/Subdivision [m] 0.16
Faces/Cross section 8.00
Render Color Margin [%] 100.00

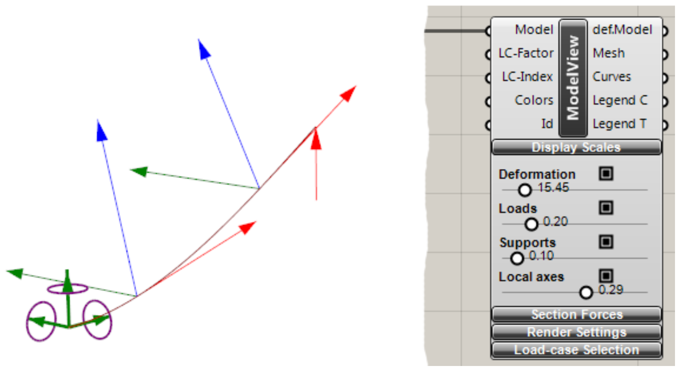
Figure 8: Color plot of strains with custom color range.

item corresponding to one element.

- The "Curve" plug delivers the axes of the deformed structure as interpolated 3^{rd} degree nurb-splines. Use the Length/Subdivision slider to set the number of interpolation points.

- "Legend C" and "Legend T" provide lists of colors and strings which can be fed into Grasshoppers Legend-component (see fig. 8). The numbers on the right side of the Legend correspond to the lower limit that the corresponding color represents. The number of color shades can be set in the "karamba.ini" file.

5.3.1 The "Display Scales"-submenu



Deformation ☑ 15.45
Loads ☑ 0.20
Supports ☑ 0.10
Local axes ☑ 0.29

Figure 9: Local axes of cantilever composed of two beam elements.

The "Display Scales"-submenu contains check boxes and sliders to enable/disable and scale displacements, load-symbols, support-symbols and local coordinate systems. The displacement scale influences the display and the output at the model-plug. It has no effect on stresses, strains, etc.. The colors of the local coordinate axes red, green, blue symbolize the local X-, Y-, and Z-axis.

5.3.2 Display of cross section forces and moments

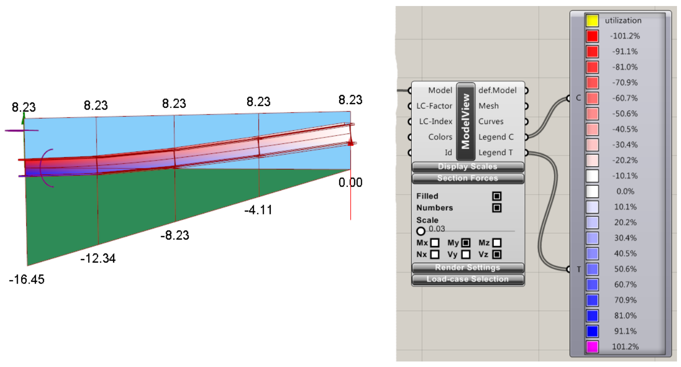


Figure 10: Moment M_y (green) about the local beam Y-Axis and shear force V_z (blue) in local Z-direction.

The section forces sub-menu lets you plot section forces and moments as curves, meshes and with or without values attached. All generated curves and meshes get appended to the ModelViews curve and Mesh output. The graphical representation is oriented according to the local coordinate axes of the beam and takes the un-deflected geometry as its base. The index of bending moments indicates the local axis about which they rotate, for shear forces it is the direction in which they act (see also fig. 21). Customize the mesh-colors vie "karamba.ini". The slider "Length/Subdivision" in sub-menu "Render Settings" controls the number of interpolation points.

5.3.3 Render settings

The "Render Settings" menu contains checkboxes for displaying different aspects of a model:

- When activated "Cross section", "Strain", "Stress" and "Utilization" result in a rendered view of the model. Utilization is the ratio between the stress at a point and the yield stress of the corresponding material. Shear is neglected for calculating strains, stresses and utilization. Do not be disappointed that the colors do not change when switching from strain to stress to utilization. The color range starts at the minimum value and stretches to the maximum. In case the model consists of one material, the zone of highest strain will also be the zone of highest stress and material utilization. Use a legend-component (see below) to get additional information out of color plots.

- "Node tags" attaches node-indexes to each node when active
- "Element ids" displays the beam identifiers
- "CroSec names" shows the cross-section name used for each beam

5.4 Supports

Without supports a structure would have the potential to freely move around in space. This is not desirable in case of most buildings. The current version of Karamba does statical calculations. This means that there must always be enough supports so that the structure to be calculated can not move without deforming. Thus rigid body modes are prohibited.

When defining the supports for a structure one has to bear in mind, that in three dimensional space a body has six degrees of freedom (DOFs): three translations and three rotations (see figure 11). The structure must be supported in such a way that none of these is possible without invoking a reaction force at one of the supports. Otherwise Karamba will refuse to calculate the deflected state. Sometimes you get results from moveable structures although you should not: The reason for this lies in the limited accuracy of computer-calculations which leads to round-off errors. Sometimes one is tempted to think that if there act no forces in one direction – consider e.g. a plane truss – then there is no need for corresponding supports. That is wrong: What counts is the possibility of a displacement.

Bad choices of support conditions constitute a source of errors that is hard to track. Later on it will be shown how to calculate the eigen-modes of a structure. This kind of calculation works also in cases of moveable structures: rigid body modes – if present – correspond to the first few eigen-modes.

Figure 12 shows the geometry developed above with supports added at the endpoints of the structure. The "Karamba/Ensemble/Support"-component takes as input either the index[3] or the coordinates of the point (or a list

[3]In order to find out the index of a specific node enable the tag-checkbox in the ModelView-component. See section 5.1.1 on how to predefine the index of specific nodes

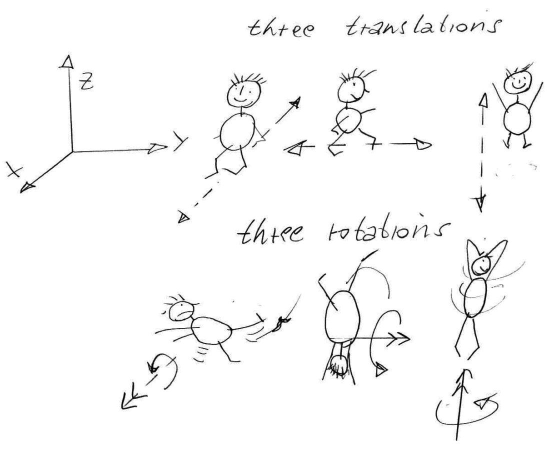
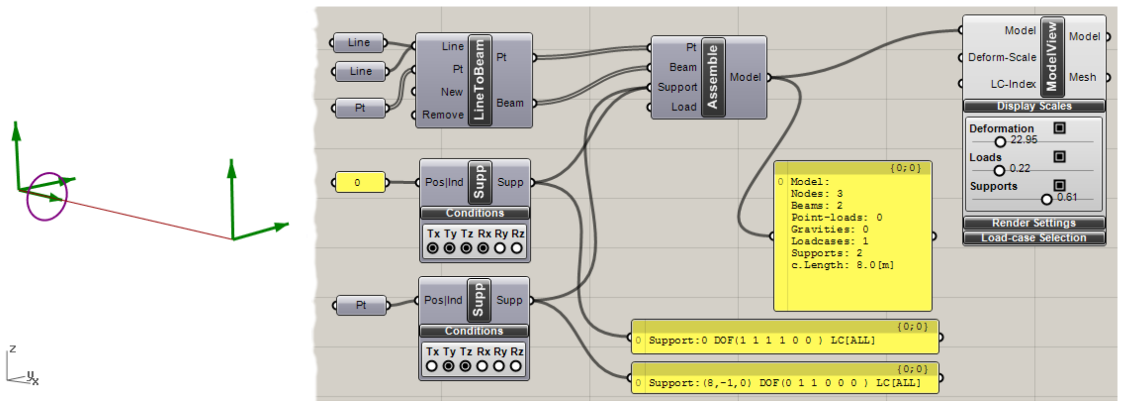


Figure 11: Metaphor for the six degrees of freedom of a body in three-dimensional space.



```
{0;0}
0 Model:
  Nodes: 3
  Beams: 2
  Point-loads: 0
  Gravities: 0
  Loadcases: 1
  Supports: 2
  c.Length: 8.0[m]
```

```
{0;0}
0 Support:0 DOF(1 1 1 1 0 0 ) LC[ALL]
```

```
{0;0}
0 Support:(8,-1,0) DOF(0 1 1 0 0 0 ) LC[ALL]
```

Figure 12: Define the position of supports by node-index or position.

with indexes or positions of points) to which it applies. Six small circles on the component indicate the type of fixation: The first three correspond to translations in global x, y and z-direction, the last three boxes stand for rotations about the global x,y and z-axis. Filled circles indicate fixation which means that the corresponding degree of freedom is zero. The state of each box can be changed by clicking on it. The string output of the component lists node-index or nodal coordinate, an array of six binaries corresponding to its six degrees of freedoms and the number of load-case to which it applies. Supports apply to all load cases by default.

From the support-conditions in figure 12 one can see that the structure is a simply supported beam: green arrows symbolize locked displacements in the corresponding direction. The translational movements of the left node

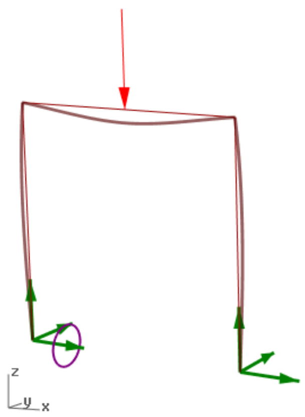
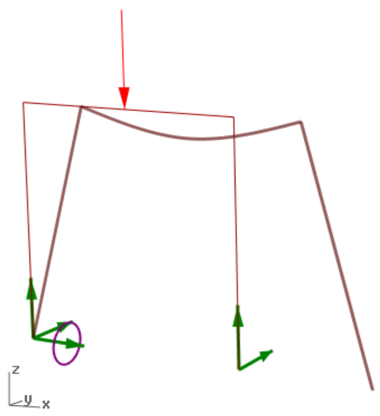
(a)  (b) 

Figure 13: Influence of support conditions – undeflected and deflected geometry. *Left:*All translations fixed at supports. *Right:* One support moveable in horizontal direction.

are completely fixed. At the right side two supports in y- and z-direction suffice to block translational movements of the beam as well as rotations about the global y- and z-axis. The only degree of freedom left is rotation of the beam about its longitudinal axis. Therefore it has to be blocked at one of the nodes. In this case it is the left node where a purple circle indicates the rotational support.

The displacement boundary conditions may influence the structural response significantly. Figure 13 shows an example for this: when calculating e.g. the deflection of a chair, support its legs in such a way that no excessive constraints exist in horizontal direction – otherwise you underestimate its deformation. The more supports you apply the stiffer the structure and the smaller the deflection under given loads. Try changing support conditions in "PortalFrame.ghx" in the examples on the karamba web-site and observe how the maximum deflection changes. In order to arrive at realistic results introduce supports only when they reliably exist.

By default the size of the support symbols is set to approximately $1.5[m]$. The slider with the heading "Support" on the ModelView-component lets you scale the size of the support symbols. Double click on the knob of the slider in order to set the range of values.

5.5 Predefined displacements

Supports as described above are a special case of displacement[4] boundary condition: They set the corresponding degree of freedom of a node to zero. The more general PreDisp-component lets you preset arbitrary displacements at nodes. Figure 14 shows a beam with prescribed, clockwise rotations at both end-points. See also example "FixedFixedBeam.ghx".

The PreDisp-component resembles the Support-component to a large degree: Nodes where displacement conditions apply can be selected via node-index[5] or nodal coordinates. The size of the list fed into the "Pos|Ind"-plug determines the number of displacement conditions. All other input will be truncated or blown up by copying its last list item.

Input-plug "LCase" lets you set the index of the load-case in which displacements shall have a specified value. The default value is "-1" which means that the displacement condition is in place for all load-cases. It is not possible to have displacement boundary active in one load-case and completely disabled in others: For load-cases not mentioned in "LCase" the PreDisp-component will act like a simple support with fixed degrees of freedom equal to zero.

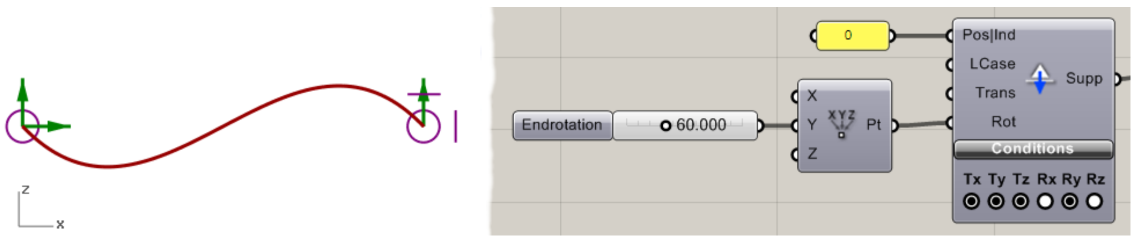


Figure 14: Left: Deflection of a beam under predefined displacements at its end-supports; Right: PreDisp-component for setting displacement condition at left support.

The "Trans"- and "Rot"-input-plugs expect vectors. They define nodal translations and rotations in global coordinates. Translations are to be given in meter, rotations in degree. The X-component of the rotation vector describes a rotation about the global X-axis. A positive value means that the node rotates counter-clockwise if the X-Axis points towards you. Analog definitions apply to rotations about the global Y- and Z-axis. Karamba is based on the assumption of small deflections. Thus be aware that large

[4]The term "displacement" as used throughout this manual includes translations and rotations.

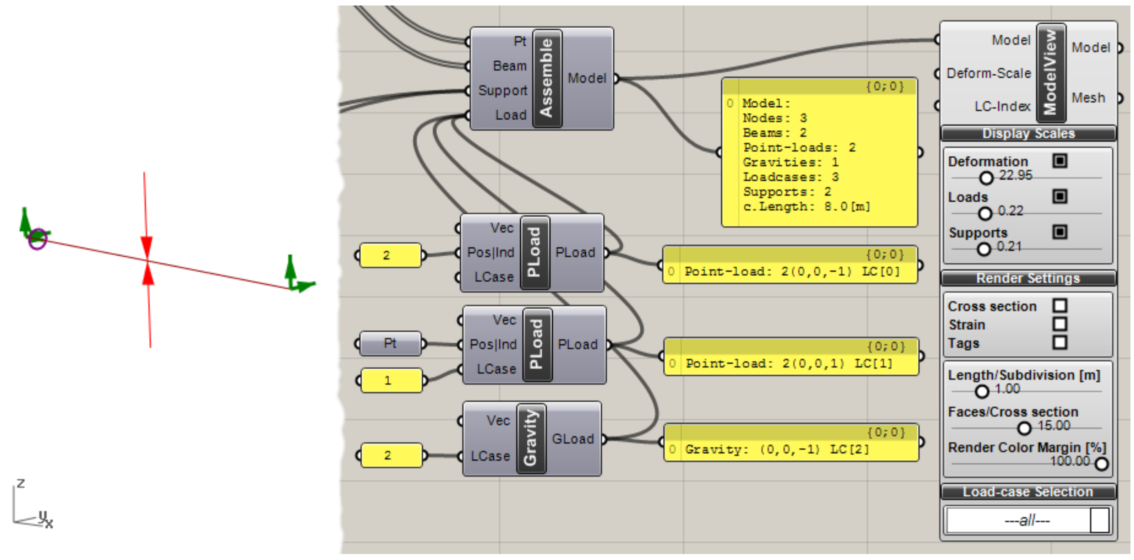
[5]In order to find out the index of a specific node enable the tag-checkbox in the ModelView-component. See section 5.1.1 on how to predefine the index of specific nodes

prescribed displacements and rotations give rise to incorrect results (which can nevertheless be used for shape-finding). In order to approximate effects due to large displacements apply them in several steps (e.g. see example "FixedFixedBeam.ghx").

Displacements can only be prescribed if the corresponding displacement degree of freedom is removed from the statical system. This means you have to click on the corresponding button in the Conditions-section of the PreDisp-component. The first three buttons stand for translations the last three for rotations. Only those components of the "Trans"- and "Rot"-vectors take effect which correspond to activated conditions.

5.6 Loads

Currently Karamba supports three kinds of loads: point-, mesh- and gravity-loads. An arbitrary number of point- or mesh-loads and one gravity-load may be combined to form a load-case of which again an arbitrary number may exist. Figure 15 shows the definition of loads with the help of Gravity- and Point-load components. On the bottom of the ModelView-component there is a drop-down-list (unfold it by clicking on the "Load-case Selection"-menu header) which can be used to select single load-cases for display. Select "–all–" in order to view all existing load-definitions of all load-cases simultaneously. Use the force-slider to scale to size of the load-symbols (double-clicking on its knob lets you change the value range and its current value).



```
{0;0}
0 Model:
  Nodes: 3
  Beams: 2
  Point-loads: 2
  Gravities: 1
  Loadcases: 3
  Supports: 2
  c.Length: 8.0[m]
```

```
{0;0}
0 Point-load: 2(0,0,-1) LC[0]
```

```
{0;0}
0 Point-load: 2(0,0,1) LC[1]
```

```
{0;0}
0 Gravity: (0,0,-1) LC[2]
```

Deformation 22.95
Loads 0.22
Supports 0.21
Cross section
Strain
Tags
Length/Subdivision [m] 1.00
Faces/Cross section 15.00
Render Color Margin [%] 100.00
---all---

Figure 15: Simply supported beam with three loads and three load-cases.

5.6.1 Point-loads

The component "Karamba/Ensemble/Point-Load" lets you define point loads. These get attached to points either by node-index[6] or node-position. Feed a corresponding list of items into the "Pos|Ind"-plug (quite analogous to the Support-component). A point-load is given as a vector. Its components define the force-components in global x-, y- and z-direction. The number of point-loads generated follows from the longest list principle [7] applied to all input. The output of the "force"-component has to be connected to the corresponding plug of the assembly. Plugging a point-load into a panel component gives the following information: Node-index where the load gets applied, force-vector and number of the load case to which it belongs.

Karamba expects all force-definitions to be in kilo Newton(kN). On earth the mass of $100kg$ corresponds to a weight force of roughly $1kN$. The exact number would be $0.981kN$ but $1kN$ is normally accurate enough. Table 1 contains the specific weight of some everyday materials. Rules of thumb numbers for loads can be found in table 2. Do not take these values too literally. For example the snow load varies strongly depending on the geographical situation.

Loads acting along lines or on a specified area can be approximated by point-loads. All you need to do is estimate the area or length of influence for each node and multiply it with the given load value. The Mesh-load-component (see next section) automates this task for surface loads.

type of material	$[kN/m^3]$
reinforced concrete	25.0
glass	25.0
steel	78.5
aluminum	27.0
fir wood	3.2
snow loose	1.2
snow wet	9.0
water	10.0

Table 1: Specific weights of some building materials

[6]In order to find out the index of a specific node enable the tag-checkbox in the ModelView-component. See section 5.1.1 on how to predefine the index of specific nodes

[7]Longest list principle means that if the input consists of lists of different length, then the longest one determines the length of all the others: They get blown up by adding their last element for as many times as required.

loads	
type	$[kN/m^2]$
life load in dwellings	3.0
life load in offices	4.0
snow on horizontal plane	1.0
cars on parking lot (no trucks)	2.5
trucks on bridge	16.7

Table 2: Loads for typical scenarios

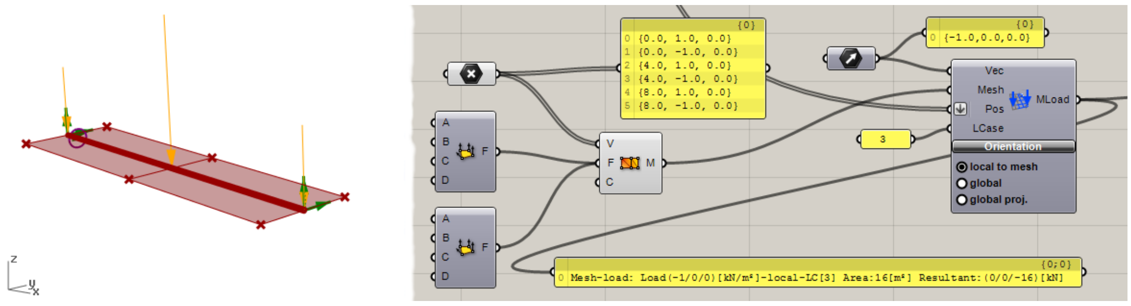
By default point loads will be put into load case zero. Any positive number fed into the LCase-plug defines the load case to which the corresponding load will be attributed. A value of -1 signals that the load acts in all existing load cases.

5.6.2 Mesh-loads

The Mesh-load-component can be used to transform surface loads into equivalent nodal loads. This lets you define life-loads on floor slabs, moving loads on bridges (see example "Bridge.ghx" in the examples collection on the Karamba web-site), snow on roofs, wind-pressure on a facade, etc... Figure 16 left side shows a simply supported beam and a mesh which consists of two rectangular faces. Each face covers one half of the beam span. The orange arrows symbolize the equivalent nodal forces. One can see that the equivalent force in the middle of the beam is twice as large as those at the supports.

The procedure for calculating nodal forces from surface loads consists of the following steps: First Karamba calculates the resultant load on each face of the given mesh. Then the resultant loads of each face get evenly distributed among their corresponding vertices. The second step consists of distributing the vertex-loads among the nodes of the structure: this is done by considering the distance between the vertices and structure-nodes where the mesh load shall apply. Each vertex transfers its load to the nearest such node. In case that there are several nodes at equal distance the vertex load gets evenly distributed among them. Differences of distance of less than $5mm$ are neglected. From the algorithm described above one can see that a crude mesh may lead to a locally incorrect distribution of loads.

The right side of figure 16 shows what data the Mesh-load-component



```
{0}
0 {0.0, 1.0, 0.0}
1 {0.0, -1.0, 0.0}
2 {4.0, 1.0, 0.0}
3 {4.0, -1.0, 0.0}
4 {8.0, 1.0, 0.0}
5 {8.0, -1.0, 0.0}
```

```
{0}
0 {-1.0,0.0,0.0}
```

```
3
```

Orientation: [✓ local to mesh] global global proj.

```
{0;0}
0 Mesh-load: Load(-1/0/0)[kN/m²]-local-LC[3] Area:16[m²] Resultant:(0/0/-16)[kN]
```

Figure 16: Simply supported beam loaded with three forces that approximate an evenly distributed surface load on a mesh.

collects: The input-plug "Vec" expects a vector or list of vectors that define the surface load. Its physical units are kilo Newton per square meter (kN/m^2). The orientation of the load-vector depends on the checkbox selected under "Orientation" (see also figure 17):

- "local to mesh": X-component of the force vector is at right angle to the mesh-face; the Y-component acts horizontally if the mesh-face X-axis is not parallel to the global Z-axis. Otherwise the Y-component of the force is parallel to the global Y-axis. This means a surface load with components only in X-direction acts like wind pressure.

- "global": The force-vector is oriented according to the global coordinate system. This makes the surface load behave like additional weight on the mesh plane.

- "global proj.": The force-vector is oriented according to the global coordinate system. The corresponding surface load is distributed on the area that results from projecting the mesh-faces to global coordinate planes. In such a way the action of snow load can be simulated.

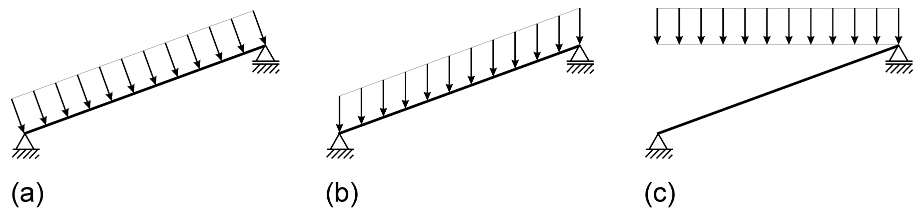


Figure 17: Orientation of loads on mesh: (a) local; (b) global; (c) global projected to global plane.

The input-plug "Mesh" accepts the mesh where the surface load shall be applied. Its vertices need not correspond to structure nodes. The mesh may have any shape.

In order to define the structure nodes where equivalent point-loads may be generated plug a list of their coordinates into the "Pos"-plug. These need to correspond with existing nodes – otherwise the Assembly-component turns red. Offending nodes will be listed in its run-time error message.

Set the "LCase"-input to the index of the load case in which the surface load shall act. Indexing of load-cases starts with zero, "-1" is short for all load cases.

For input plugs "Vec", "Mesh" and "LCase" the longest list principle applies.

5.6.3 Gravity-loads

Each load case may contain zero or one definition for the vector of gravity. In this way one can e.g. simulate the effect of an earthquake by applying a certain amount of gravity in horizontal direction. For Vienna which has medium earthquake loads this amounts to approximately 14% of gravity that a building has to sustain in horizontal direction. In areas with severe earthquake loads this can rise to 100%.

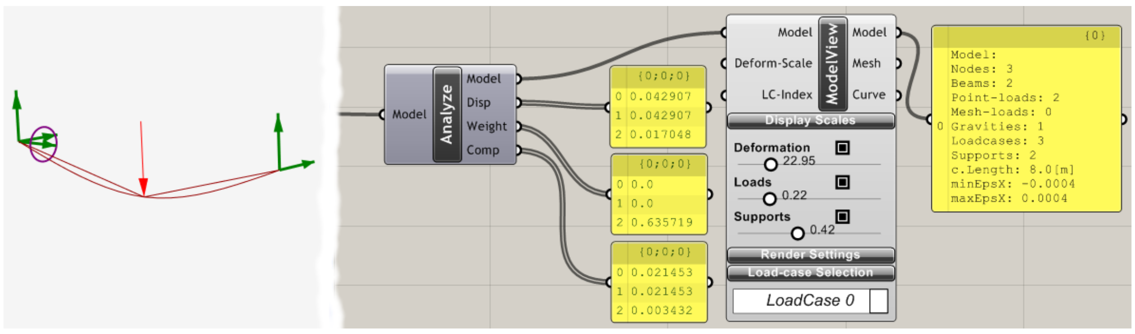
The gravity component applies to all active beams in the statical model. The gravity vector defines the direction in which gravity shall act. A vector of length one corresponds to gravity as encountered on earth.

5.7 Analysis

With geometry, supports and loads defined the statical model is ready for further processing. The Analysis-component computes the deflection for each load case and adds this information to the model. Whenever the Analysis-component reports an error (turns red) despite the fact that the Assemble component works, it is probably a good idea to check the support conditions.

Figure 18 shows a deflected beam. The analysis component not only computes the model deflections but also outputs the maximum nodal displacement (in meter),the maximum total force of gravity (in kilo Newton) and the structures internal deformation energy from each load case - see section 5.8.2 for details on work and energy.

These values can be used to rank structures in the course of a structural optimization procedure: the more efficient a structure the smaller the max-



```
{0;0;0}
0 0.042907
1 0.042907
2 0.017048
```

```
{0;0;0}
0 0.0
1 0.0
2 0.635719
```

```
{0;0;0}
0 0.021453
1 0.021453
2 0.003432
```

Deformation 22.95
Loads 0.22
Supports 0.42
LoadCase 0

```
{0}
  Model:
  Nodes: 3
  Beams: 2
  Point-loads: 2
  Mesh-loads: 0
0 Gravities: 1
  Loadcases: 3
  Supports: 2
  c.Length: 8.0[m]
  minEpsX: -0.0004
  maxEpsX: 0.0004
```

Figure 18: Deflection of simply supported beam under single load in mid-span.

imum deflection, the amount of material used and the value of the internal elastic energy. Real structures are designed in such a way that their deflection does not impair their usability. See section 5.12 for further details. Maximum deflection and elastic energy both provide a benchmark for structural stiffness yet from different points of view: the value of elastic energy allows to judge a structure as a whole; The maximum displacement returns a local peak value.

When activating the preview property of the Analysis-component the undeflected structure shows up. In order to view the deflected model use the ModelView-component and select the desired load case in the menu "Load case Selection". There exist two options for scaling the deflection output. First there is a slider entitled "Deformation" in the menu "Display Scales" that lets you do quick fine-tuning on the visual output. Second option: the input-plug "LC-Factor" which accepts a list of numbers that ModelView uses to scale the loads. Its default value is 1.0. Each item in the list applies to a load case. If the number of items in this list and the number of load cases do not match then the last number item is copied until there is a one to one correspondence. The second option for scaling displacements can be used in the course of form-finding operations: The model-plug at the right side of the ModelView outputs the displaced geometry which can be used for further processing. Selecting item "–all–" on the drop-down-list for the selected load case results in a superposition of all load cases with their corresponding scaling factor.

Looking at figure 18 one immediately notices that only beam center axes are shown. In order to see the beams in a rendered view activate the "Cross section"-checkbox on the ModelView-component in Menu. This results in an image such as in figure 19(a). The mesh of the rendered image is available

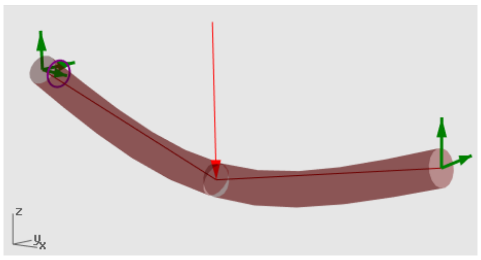
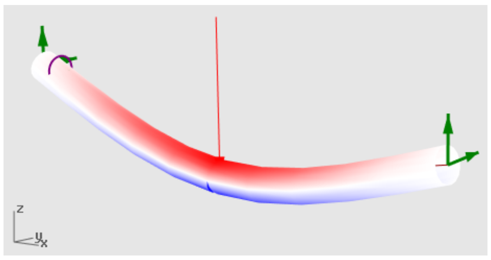
(a)  (b) 

Figure 19: Rendered images of the beam. *Left:* Only deflections enabled. *Right:* Deflections and strains enabled.

at the "Mesh"-output of the model view. Two sliders control the mesh-size of the rendered beams: First "Length/Subdivision" determines the size of sections along the middle axis of the beams. Second "Faces/Cross section" controls the number of faces per cross-section.

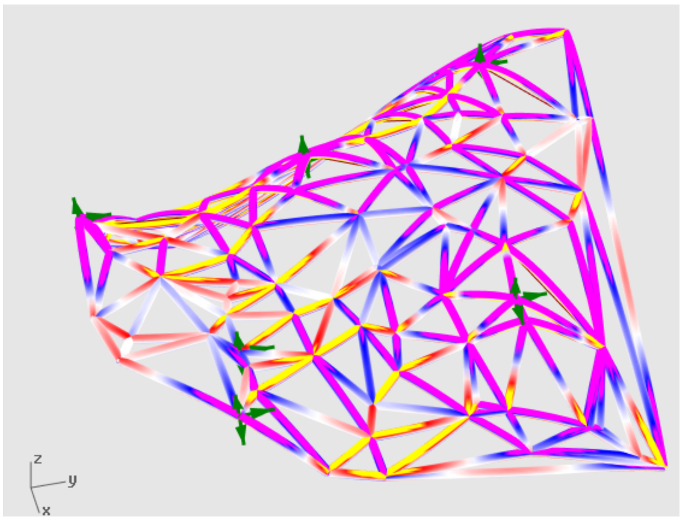


Figure 20: Mesh of beams under dead weight with Render Color Margin set to 5%.

It is instructive to see which parts of a beam are under tension or compression. Activate the "Strain"-checkbox in menu "Render Settings" in order to display the strains in longitudinal beam direction. Red (like brick) means compression, blue (like steel) tension. Strain is the quotient between the increase of length when loaded and the initial length of a piece of material (compressive strain is negative, tensile strain positive). In some models there may exist small regions with high strains with the rest of the structure having comparatively low strain levels. This results in a strain rendering that is predominantly white and not very informative. With the slider "Render Color Margin" of the "Render Settings Menu" you can set the percentage of maximum tensile and compressive strain at which the color-scale starts. Compressive strain values beyond that level appear yellow, excessive tensile strains pink (see figure 20).

5.8 Results

5.8.1 Section forces

The S-Force component retrieves axial forces N and resultant bending moments M for all elements and load cases. See fig. 21 for the definition of N and M. The order of element results corresponds to the order of beams. Thus the data can be used for cross section design.

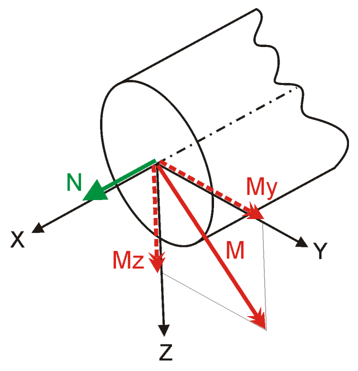
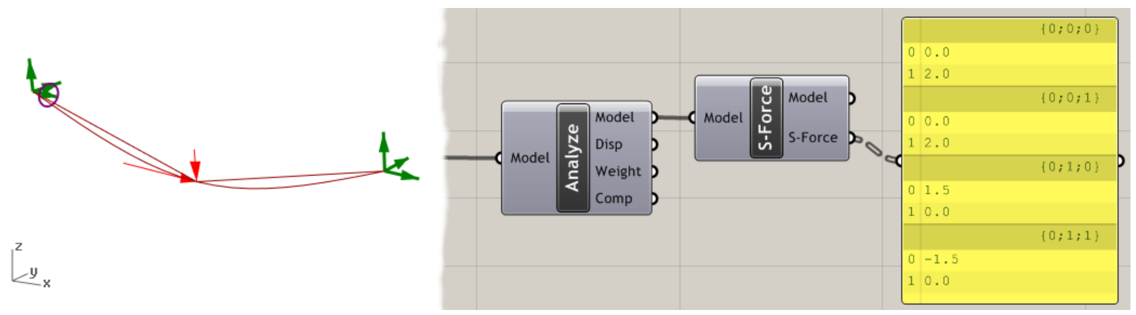


Figure 21: Normal force N and resultant moment M at cross section with local coordinate axes XYZ.

Figure 22 shows a simply supported beam with two load cases presented in one picture. The beam consists of two elements and has a total length of eight meters. In load case zero a vertical force of magnitude $1kN$ acts vertically downwards in the middle of the beam. Load case one consists of a point-load of $3kN$ directed parallel to the un-deformed beam axis. The results at the output-plugs "N" and "M" in fig. 22 are a three-dimensional trees that hold the beams Normal force in kilo Newton [kN] and resultant bending Moment in kilo Newton times meter [kNm] respectively. There is only one model fed into the S-Force component thus the first index is zero. The second index refers to the load case: the first two lists contain results for load case zero, the last two for load case one. Index three corresponds to the element indices in the model.

Tensile normal forces come out positive, compressive normal forces have negative sign. The resultant moment yields always positive values as it is the length of the resultant moment vector in the plane of the cross section.



```
{0;0;0}
0 0.0
1 2.0
{0;0;1}
0 0.0
1 2.0
{0;1;0}
0 1.5
1 0.0
{0;1;1}
0 -1.5
1 0.0
```

Figure 22: Simply supported beam under axial and transversal point-load: List of Normal forces and moments for all elements and all load cases.

Karamba currently computes section forces at the endpoints of elements and returns their maximum values. In case of zero gravity the maximum values of M and N occur at the endpoints. With gravity switched on these maxima may lie inside the elements. In order to get a good approximation of the maximum cross section forces divide the elements in little pieces. As M is always rendered positive the maximum at the end points is unambiguously given. Under gravity normal forces in a beam may change sign. In such a case Karamba returns that N which gives the maximum absolute value.

Let us take a look at the output in fig. 22. In load case zero both elements return zero normal force because there acts no external axial load. The maximum moment of both elements is $2[kNm]$. For a simply supported beam under a mid-point transverse load the maximum moment occurs in the middle and turns out to be $M = F \cdot L/4 = 1[kN] \cdot 8[m]/4 = 2[kNm]$.

The axial force of $3[kN]$ in load case one flows to equal parts into both axial supports. It causes tension $(1.5[kN])$ in the left element and compression $(-1.5[kN])$ in the right one.

5.8.2 Elastic energy

In mechanics energy is equal to force times displacement parallel to its direction. Think of a rubber band: if you stretch it you do work on it. This work gets stored inside the rubber and can be transformed into other kinds of energy. You may for example launch a small toy airplane with it: then the elastic energy in the rubber gets transformed into kinetic energy. When stretching an elastic material the force to be applied at the beginning is zero and then grows proportionally to the stiffness and the increase of length of the material. The mechanical work is equal to the area beneath the curve that results from drawing the magnitude of the applied force over its corresponding displacement. In case of linear elastic materials this gives a rectangular triangle with the final displacement forming on leg and the final force being its other leg. From this one can see, that for equal final forces the elastic energy stored in a material decreases with decreasing displacements which corresponds to increasing stiffness.

The structure of the results list returned from the E-Energy component resembles that of the S-Force component described above. Instead of normal force and moment the work of normal force and moment on each beam are given.

5.8.3 Nodal displacements

The "Disp" component lists the displacements of each node for all load cases. Again two trees with three dimensions form its output. These dimensions correspond to Model/LoadCase/Node. The data for each node at the output plugs "Trans" and "Rot" consists of three values each: three translations and three rotations. All of them are given in the global coordinate system in meter and radiant. A positive rotation say about the global X-axis means that the node rotates counter clockwise for someone who looks at the origin of the coordinate system and the X-axis points towards him or her.

5.9 How to change beam properties

By default Karamba assumes the cross-section of beams to be steel tubes with a diameter of $10[cm]$ and a wall-thickness of $0.33[cm]$. When two beams meet they are rigidly connected like they were welded together. Use the ModifyBeam-component to set the beam properties according to your choice. Figure 23 shows how this can be done by inserting it in front of the Assemble-component. By default the ModifyBeam-component leaves all incoming beams unchanged. Negative values for input properties take no effect. The size of the lists of input data is scaled to match the number of input beams by copying their last item. Several ModifyBeam-components may act consecutively on the same beam.

5.9.1 Bending stiffness

Beams resist normal force and bending. Setting the "Bending"-plug of the ModifyBeam-component to false disables bending stiffness and turns the

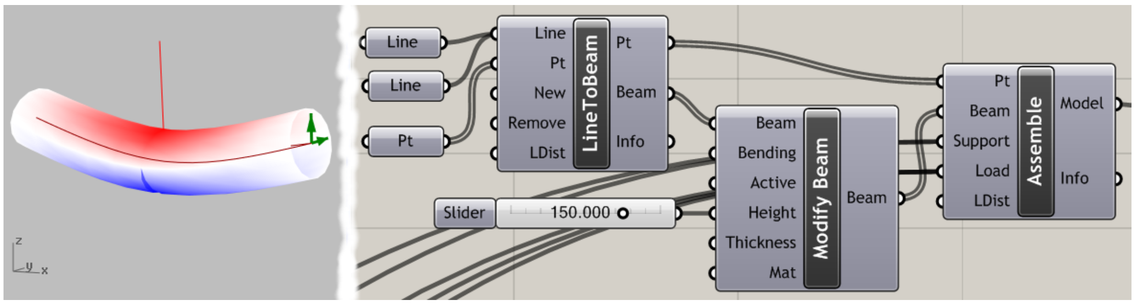


Figure 23: Modification of the default beam properties.

corresponding beam into a truss. There exist reasons that motivate such a step:

- Connections between beams that reliably transfer bending and normal force are commonly more expensive than those that carry normal force only. The design of connections heavily depends on the kind of material used: rigid bending connections in wood are harder to achieve than in steel. Yet rigid connections add stiffness to a structure and reduce its deflection. Therefore you are always on the safe side if you use truss elements instead of beams.

- For beams with small diameter compared to their length the effect of bending stiffness is negligible compared to axial stiffness. Just think of a thin wire that is easy to bend but hard to tear by pulling.

- Abandoning bending stiffness reduces computation time by more than half for each node with only trusses attached.

- Karamba bases deflection calculations on the initial, undeformed geometry. Some structures like ropes are form-active. This means that when a rope spans between to points the deformed geometry together with the axial forces in the rope provide for equilibrium. This effect is not taken into account in Karamba. In Karamba only the bending stiffness of the rope (which is very small) keeps it from deflecting indefinitely. One way to circumvent this lies in using a truss instead of a beam-element. The second possibility would be to reduce the specific weight of the rope to zero (see further below). The third possibility would be to start from a slightly deformed rope geometry and apply the external loads in small steps where the initial geometry of each step results from the deformed geometry of the previous one.

Trusses only take axial forces. Therefore they do not prevent the nodes they are connected to from rotating. In case that only trusses attach to a node, Karamba automatically removes its rotational degrees of freedom. Otherwise the node could freely rotate which is a problem in static calculations. As soon as one beam connects to a node the node has rotational degrees of freedom. Bear this in mind when the Analysis-component turns red and reports a kinematic system. Transferring only axial forces means that a truss reduces a nodes movability in one direction. A node that is not attached to a support has three translational degrees of freedom. Thus there must be three truss elements that do not lie in one plane for a node to be fixed in space.

5.9.2 Activation status of beams

When set to true this option excludes the corresponding beam from further calculations until it is reset to true.

5.9.3 Height and wall-thickness of cross-sections

Height – which in case of circular tubes is equivalent to the outer diameter D – and wall-thickness of a cross-section determine a beams axial and bending stiffness. Karamba expects both input values to be given in centimeter. The cross-section area is linear in both diameter and thickness whereas the moment of inertia grows linearly with thickness and depends on D^3. So in case of insufficient bending stiffness it is much more effective to increase a beams height (or diameter) than increasing its wall thickness. If a cross-section hight D is given but no wall thickness t then Karamba assumes $t = D/30$ by default.

5.9.4 Changing the default material

Steel is the default material of all elements. The default grade is S235 which means that it yields at $f_y = 23.5[kN/m^2]$.The next section shows how to define materials with arbitrary properties or how to select predefined materials from a data base. Feed these into the "Mat" input-plug of the "ModifyBeam"-component in order to override the default values.

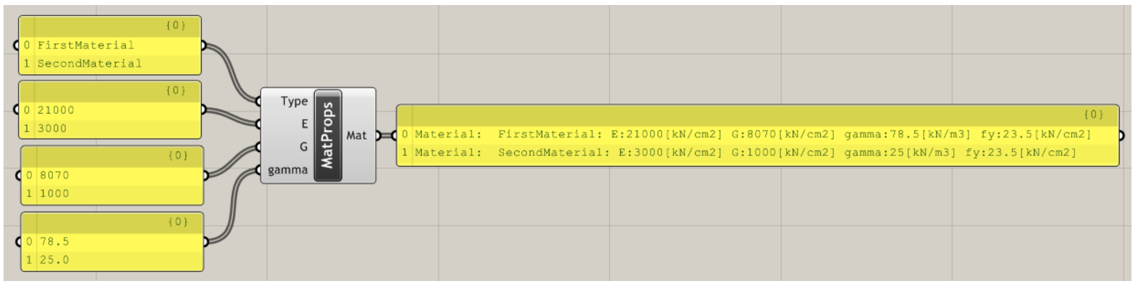
5.10 Materials

There are two ways for defining a material in Karamba: Either select a material by type from a data-base (see section 5.11) or set each material property manually (see below).

5.10.1 Material properties definition

The component "MatProps" lets one directly define material properties (see fig. 24). These are

- Young's Modulus "E"
- Shear modulus "G"
- specific weight "gamma"
- yield stress "fy"



```
0 Material:  FirstMaterial: E:21000[kN/cm2] G:8070[kN/cm2] gamma:78.5[kN/m3] fy:23.5[kN/cm2]
1 Material:  SecondMaterial: E:3000[kN/cm2] G:1000[kN/cm2] gamma:25[kN/m3] fy:23.5[kN/cm2]
```

Figure 24: The definition of the properties of two materials via the MatProps component.

In addition to these, user defined materials can be given a name via the input-plug "Name" for identification in later stages of calculation.

5.10.2 Material stiffness

The stiffness i.e. resistance of a material against deformation is characterized by its Young's Modulus or modulus of elasticity E. The higher its value the stiffer the material. Table 3 lists E-values for some popular building materials.

For composite materials – like in the case of rods made from glass fiber and epoxy – it is necessary to defer a mean value for E by material tests. Karamba expects the input for E to be in kilo Newton per square centimeter $[kN/cm^2]$.

type of material	$E[kN/cm^2]$
steel	21000
aluminum	7000
reinforced concrete	3000
glass fiber	7000
wood (spruce)	1000

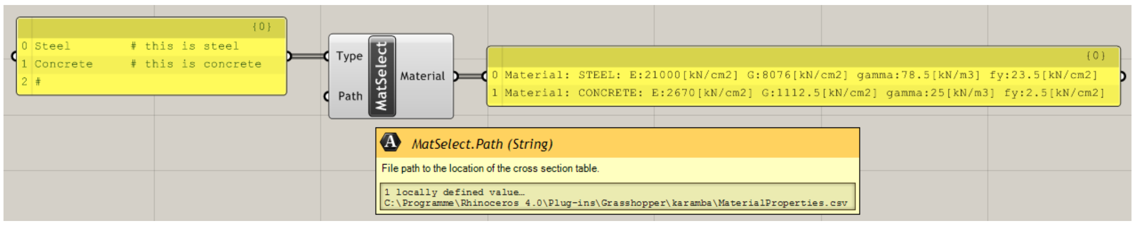
Table 3: Young's Modulus of materials

If one stretches a piece of material it not only gets longer but also thinner: it contracts laterally. In case of steel for example lateral strain amounts to 30% of the longitudinal strain. In case of beams with a large ratio of cross section height to span this effect influences the displacement response. In common beam structures however this effect is of minor importance. The shear modulus "G" describes material behavior in this respect. It is included in the "MaterialProps" component for completeness. Currently Karamba does not take account of this property.

5.10.3 Specific weight

The value of "gamma" is expected to be in kilo Newton per cubic meter $[kN/m^3]$. This is a force per unit of volume. Due to Earths gravitational acceleration ($a = g = 9.81[kg \cdot m/s^2]$) and according to Newtons law ($f = m \cdot a$) a mass m of one kilogram acts downwards with a force of $f = 9.81N$. For calculating deflections of structures the assumption of $f = 10N$ is accurate enough. Table 1 gives specific weights of a number of typical building materials. The weight of materials only takes effect if gravity is added to a load case (see section 5.6.3).

5.11 Material selection


```
{0}
0 Steel         # this is steel
1 Concrete      # this is concrete
2 #
```

```
{0}
0 Material: STEEL: E:21000[kN/cm2] G:8076[kN/cm2] gamma:78.5[kN/m3] fy:23.5[kN/cm2]
1 Material: CONCRETE: E:2670[kN/cm2] G:1112.5[kN/cm2] gamma:25[kN/m3] fy:2.5[kN/cm2]
```

MatSelect.Path (String)

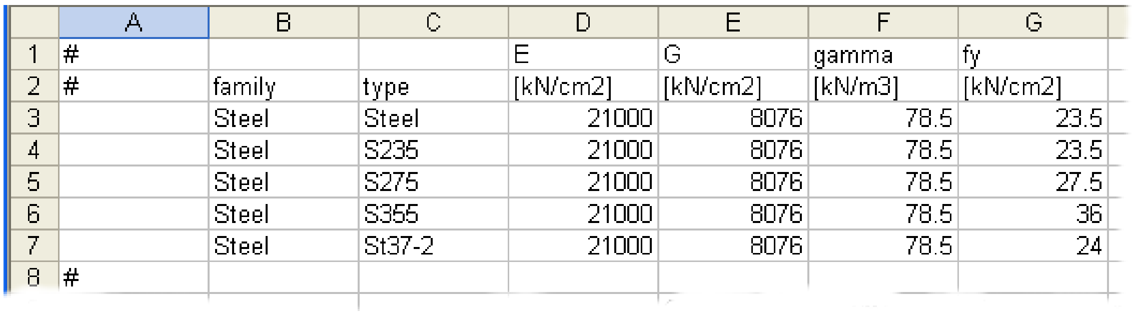
File path to the location of the cross section table.

```
1 locally defined value…
C:\Programme\Rhinoceros 4.0\Plug-ins\Grasshopper\karamba\MaterialProperties.csv
```

Figure 25: Selection of material definitions by type.

The "MatSelect"-component in the menu subsection "Material" lets you select materials by name via the input-plug "Name" which expects a string or list of strings (see fig. 25). The data-base currently holds properties for 'steel', 'concrete', 'wood' and 'aluminum'. Upper and lowercase letters may be arbitrarily mixed, spaces get removed prior to searching the data-base. A '#' means that the rest of the line is comment.

There exist different types of steel, concrete etc.. The generic term 'concrete' will result in the selection of an everyday type of concrete - a C25/30 according to Euro-code. More specific descriptions may be given: Have a look at the data-base in order to get an overview. This data-base by default resides in "...Rhinoceros 4.0/Plug-ins/Grasshopper/Karamba/-MaterialProperties.csv". The extension .csv stands for "comma separated value". The file can be opened with any text editor and contains the table entries separated by semicolons. It is preferable however to use OpenOffice or Excel (both can read and write csv-files) because they render the data neatly formatted (see fig. 26). Make sure to have a "." and not a "," set as your decimal separator. The setting may be changed under Windows via "regional settings" in "system settings". All lines in the table that start with "#" are comments. Feel free to define your own materials.



	A	B	C	D	E	F	G
1	#			E	G	gamma	fy
2	#	family	type	[kN/cm2]	[kN/cm2]	[kN/m3]	[kN/cm2]
3		Steel	Steel	21000	8076	78.5	23.5
4		Steel	S235	21000	8076	78.5	23.5
5		Steel	S275	21000	8076	78.5	27.5
6		Steel	S355	21000	8076	78.5	36
7		Steel	St37-2	21000	8076	78.5	24
8	#						

Figure 26: Partial view of the default data base underlying the MatSelect component

The file path to the materials data-base can be changed in two ways: first right-click on the component and hit "Select file path to material definitions" in the context menu that pops up. Second plug a panel with a file path into "Path". Relative paths are relative to the directory where your definition lies.

5.11.1 Theoretical background of stiffness, stress and strain

As mentioned in section 5.7 strain is the quotient between the increase of length of a piece of material when loaded and its initial length. Karamba can visualize strains as color plots - in order to do this activate the strain-checkbox in the ModelView-component. Usually one uses the greek letter ε for strains. Strain induces stress in a material. Stress has the meaning of force per unit of area. From the stress in an beam cross-section one can calculate the normal force that it withstands by adding up (integrating) the product of area and stress in each point of the cross-section. Stress is normally symbolized by the greek letter σ. Linear elastic materials show a linear dependence between stress and strain. The relation is called Hooke's Law and looks like this:

$$\sigma = E \cdot \varepsilon$$

E stands for Young's Modulus which depends on the material and depicts its stiffness. Hooke's law expresses the fact that the more you deform something the more force you have to apply.

5.12 Tips for designing statically feasible structures

Karamba can be used to analyze the response of structures of any scale. It is based on two assumptions: First deflections are small as compared to the size of the structure. Second materials do behave in a linear elastic manner – i.e a certain increase of deformation is always coupled to the same increase of load. Real materials behave differently: they weaken at some point and break eventually. If you want to calculate structures with large deflections you either have to increase the load in several steps and update the deflected geometry (as described in section 5.9.1) or live with the fact that the results somewhat deviate from the real response.

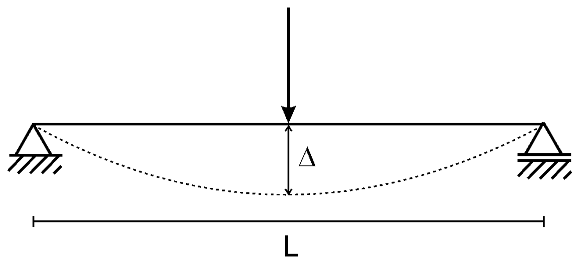


Figure 27: Simply supported beam.

For typical engineering structures the assumptions mentioned above suffice for an initial design. In order to get meaningful cross section dimensions limit the maximum deflection of the structure.

Figure 27 shows a simply supported beam of length L with maximum deflection Δ under a single force at mid-span. The maximum deflection of a building should be such that people using it do not start to feel uneasy. As a rough rule of thumb try to limit it to $\Delta \leq L/300$. If your structure is more like a cantilever $\Delta \leq L/150$ will do. This can always be achieved by increasing the size of the cross-section. If deflection is dominated by bending (like in figure 27) it is much more efficient to increase the height of the cross-section than its area (see section 5.9.3). Make sure to include all significant loads (dead weight, live load, wind...) when checking the allowable maximum deflection. For a first design however it will be sufficient to take a multiple of the dead-weight (e.g. with a factor of 1.5). This can be done in Karamba by giving the vector of gravity a length of 1.5.

In case of structures dominated by bending, collapse is preceded by large deflections (see for example the video of the collapse of the Tacoma-Narrows bridge at `http://www.youtube.com/watch?v=3mclp9QmCGs` which also gives an impression of what normal shapes are; see also example "Bridge.ghx" with the EModes-component enabled). So limiting deflection automatically leads to a safe design. If however compressive forces initiate failure, collapse may occur without prior warning. The phenomenon is called buckling. In Karamba it makes no difference whether an axially loaded beam resists compressive or tensile loads: it either gets longer or shorter and the absolute value of its change of length is the same. In real structures the more slender a beam the less compressive force it takes to buckle it. An extreme example would be a rope. As a rule of thumb limit the slenderness – which is approximately the ratio of free span to diameter – of compressed elements to $1/100$.

5.13 Cross sections

Karamba offers four basic types of cross section:

- circular tube – the default
- hollow box section
- filled trapezoid section
- I-profile

The dimensions of each of these may be defined manually or by reference to a cross section table (see section **??**).

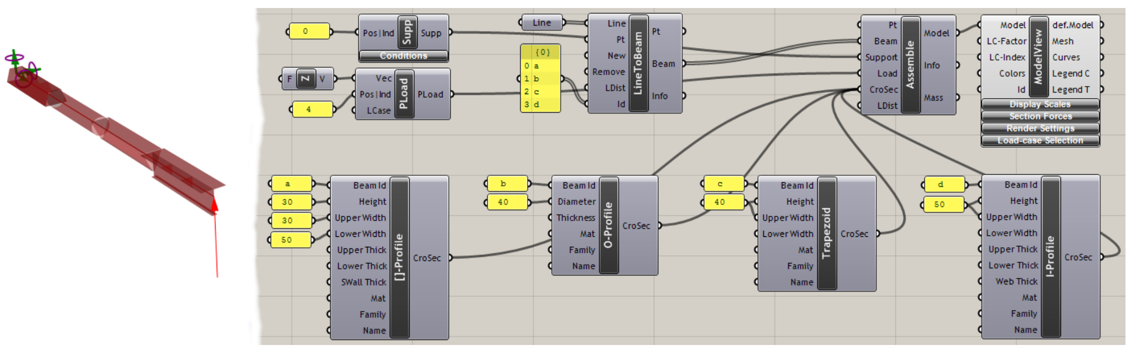


Figure 28: Cantilever with four different kinds of cross section.

For each type of cross section there exist two ways of attaching them to beams:

- The first works analogously to the "ModifyBeam"-component: on the left side it takes a list of beams as input, attaches a cross section of given properties and output them on the right side.
- Alternatively cross sections can be defined as autonomous entities which may be plugged into the "Assemble" component (see fig. 28). They know about the beams they belong to by their "Beam Id" property: This is a list of strings containing beam identifiers or regular expressions. Upon assembly all beam identifiers are compared to all "Beam Id" entries of a cross section. In case of correspondence the cross section is attached to the beam. So this second method for attaching cross sections to beams overrules the first method.

Components of the first kind have names ending with "-Beam", those of the second kind have names with '-Profile" at their back.

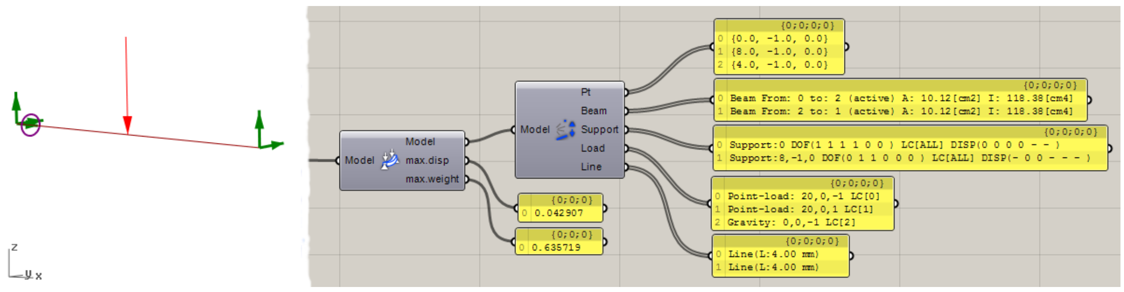
5.14 Cross section properties definition

Fig. 28 shows a cantilever with cross section properties defined by means of beam identifiers. The beam axis always coincides with the center of gravity of a cross section. Changing e.g the upper flange width of an I-section therefore results in a slight movement of the whole section in the local Z-direction.

Apart from the input-plugs that define the cross section geometry there are the "Mat"-, "Family"- and "Name"-plug:

- "Mat" accepts a material definition. It defaults to Steel "S235".
- "Family": Each cross section belongs to a family. When doing cross section optimization Karamba selects only profiles that belong to the same family as the original section.
- "Name": an identifier, supposed to be unique for each cross section. Enable "CroSec names" in ModelViews "RenderSettings"-submenu in order to view them.

5.15 The Dissemble-component



```
{0;0;0;0}
0 {0.0, -1.0, 0.0}
1 {8.0, -1.0, 0.0}
2 {4.0, -1.0, 0.0}
```

```
{0;0;0;0}
0 Beam From: 0 to: 2 (active) A: 10.12[cm2] I: 118.38[cm4]
1 Beam From: 2 to: 1 (active) A: 10.12[cm2] I: 118.38[cm4]
```

```
{0;0;0;0}
0 Support:0 DOF(1 1 1 1 0 0 ) LC[ALL] DISP(0 0 0 0 - - )
1 Support:8,-1,0 DOF(0 1 1 0 0 0 ) LC[ALL] DISP(- 0 0 - - - )
```

```
{0;0;0;0}
0 Point-load: 20,0,-1 LC[0]
1 Point-load: 20,0,1 LC[1]
2 Gravity: 0,0,-1 LC[2]
```

```
{0;0;0;0}
0 Line(L:4.00 mm)
1 Line(L:4.00 mm)
```

```
{0;0;0}
0 0.042907
```

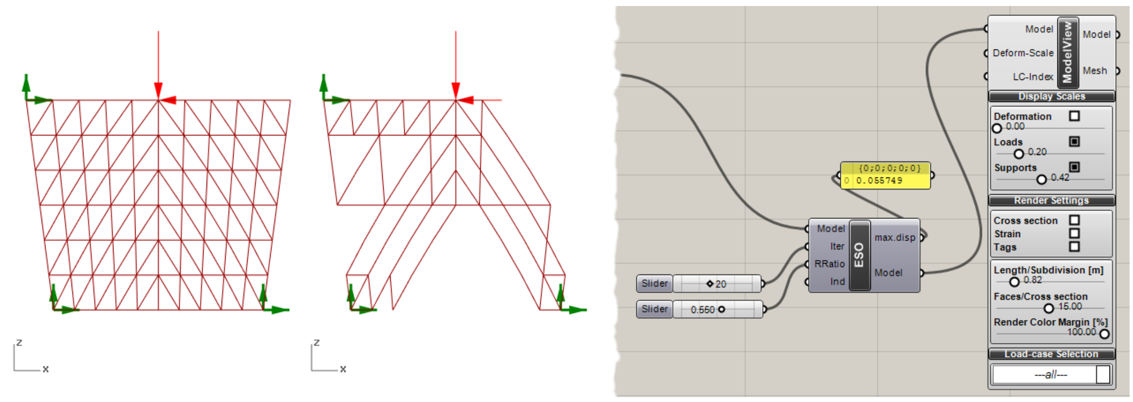
```
{0;0;0}
0 0.635719
```

Figure 29: Model is dissolved into its ingredients.

It is sometimes necessary to put apart existing models in order to reassemble them in different configurations. The Dissemble-component can be used for dissolving a statical model into its ingredients (see figure 29). Be aware of the fact that index references of loads or supports to nodes will remain unaltered. Mixing them with incompatible geometries results in errors or unexpected results. The lines-output plug lists only lines that coincide with active elements. This allows for example to extract the structural parts that remain after evolutionary structural optimization (see below).

5.16 Evolutionary structural optimization

Evolutionary structural optimization (ESO) constitutes a method of topology optimization which was pioneered by Y.M. Xie and G.P. Steven. The underlying principle is simple: One starts from a given volume made up from structural elements on predefined supports and with preset loads acting on it. Calculating the structural response will show that there are regions which carry more of the external load than others. Now one removes a number of those elements that are least strained and thus least effective in the structure. Again the response of the now thinned out structure is determined and under-utilized elements removed and so on. This iterative procedure stops when a target volume or number of remaining structural elements is reached.



Deformation 0.00
Loads 0.20
Supports 0.42
Cross section
Strain
Tags
Length/Subdivision [m] 0.82
Faces/Cross section 15.00
Render Color Margin [%] 100.00
---all---
{0;0;0;0;0}
0 0.055749
Slider 20
Slider 0.550

Figure 30: Triangular mesh of beams before (left) and after application of ESO.

Karamba uses the absolute value of normal force of the beams as an indicator of their effectiveness. This means that it renders incorrect results in case of structures dominated by bending. Figure 30 shows the ESO-component (which can be found in the subsection "algorithms") at work. On the left side one can see the initial geometry which is a triangular mesh derived from a surface (see example "ESOWall.ghx" for details). There exist two load cases with loads acting in the plane of the structure in horizontal and vertical direction respectively. Three corner nodes of the structure are held fixed. The right picture shows the optimized structure with 55% of initial beams removed in the course of 20 design iterations. The ESO-component expects a model to be plugged into its left side. The number if design iterations "Iter" determines the number of design iterations: the higher this number the less beams will be removed per iteration and the longer the computation time. The ratio of the number of beams to be removed to the initial number of elements is set by the value in "RRatio". Its value is expected to be $0 \leq RRatio \leq 1$. Sometimes one only wants to optimize certain regions of a structure. In such a case on can feed a list of beam indexes into the "Ind"-plug of the ESO-component. This will limit the application of the ESO-procedure to these elements. If no input is given here, then all beams of a model will be included in optimization by default.

Karamba implements the ESO-method in a basic way: The only criteria for element removal is the mean value of the axial force of the beams over all load cases. If the number of iterations is selected too low then it may occur that single beams get disconnected from the main structure and they seem to fly. The reason for this lies in the fact that Karamba applies a so called soft-kill approach for thinning out the structure: elements are not removed but simply given small stiffness values. This ensures that structural response can be calculated under all circumstances. At the end of the ESO procedure those beams that were given small stiffness are set inactive. This may cause kinematic structures if unconnected beams exist in the optimized result.

5.17 Bi-directional evolutionary structural optimization

The bidirectional evolutionary structural optimization (BESO)[8] method carries the ideas behind ESO one step further: Instead of always removing structural parts it also reactivates elements during optimization. This takes account of the fact, that an element which got removed in an early stage of optimization may gain importance later on.

The default procedure as used in karambas "BESO"-component consists of the following steps:

1. Deactivate all participating elements by setting their Young's Modulus to a very small value.
2. Calculate the structural response for given loads and rate the elements according to a user defined criteria.
3. Activate a predefined number of the highest ranking elements.
4. Recalculate the structural response and rank elements.
5. Deactivate a predefined number of the lowest ranking elements.
6. If the target number of active elements is reached stop the iteration otherwise proceed to step two.

Figure 31 shows the BESO-component. Here the meaning of each input parameter:

"Model" : receives the model to be optimized.

"Iter" : the target number of BESO-iterations: the more iterations the better the results and the more time-consuming the optimization.

[8]This component was devised and programmed by Robert Vierlinger. The following section is based on his written explanations.

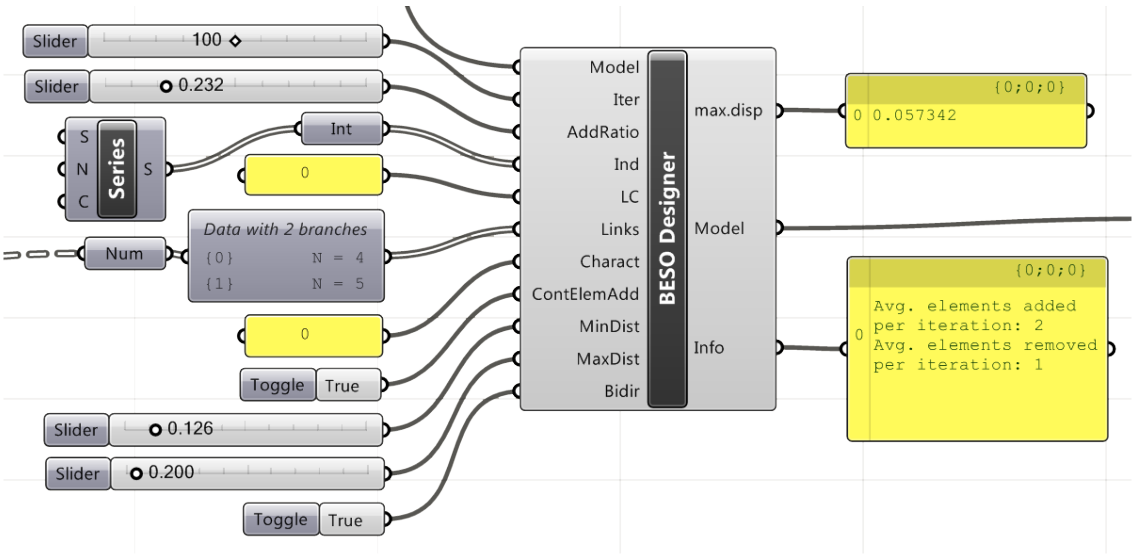


Figure 31: The BESO component in action.

"Ratio" : the target ratio of active elements to all participating elements. The resulting structure may slightly deviate from this ratio. This is due to additional boundary conditions like grouping of elements or distance relations between added or removed elements which can be imposed by the user.

"Ind" : Indices of elements that take part in the BESO design process. If none are specified, BESO is carried out for the entire model.

"LC" : index of load case to be considered. Zero is the index of the first load case.

"Links" : Sometimes it is desirable to remove or add patches of elements as a whole. Feed a two-dimensional tree into the "Links"-plug: The element indices in each leaf define one group. Activation and deactivation of groups works as follows: If a group contains an element that qualifies for activation, the entire group is added. In case of group elements that qualify for removal the ranking value of all other group members is checked. The group stays active as long as all other elements ranking value is twice as high as the current limit value for deactivation.

"Charact" : Sets the element ranking criteria by number as described below:

- "0": absolute value of normal force
- "1": bending energy

- "2": deformation energy stored in an element which is equal to the sum of bending and axial deformation energy
- "3": specific deformation energy (i.e. elastic energy stored in an element per unit of volume; this prevents big elements from outscoring small ones)

"ContElemAdd" : In some cases it makes sense to activate only such elements that connect to either supports or previously added elements. Set "ContElemAdd" to true to enable that option.

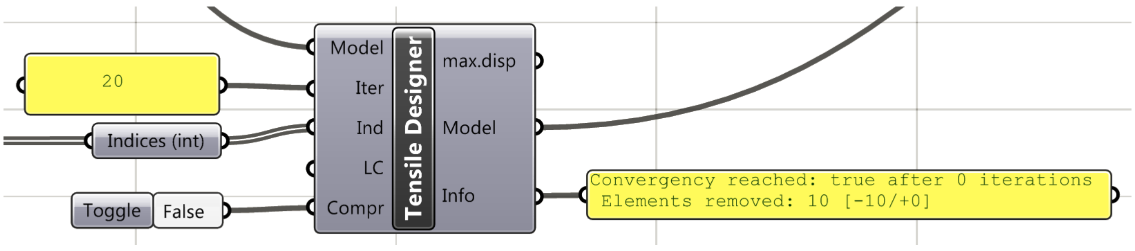
"MaxDist" : sets the upper threshold distance for the end-points of neighboring elements to count as connected if option "ContElemAdd" is enabled.

"MinDist" : sets a limit on the minimum distance between the end-points of active and elements to be newly added. This prevents clustering and spatial overlap between elements.

"BiDir" : if set to "False" the algorithm performs pure growth until the desired ratio of active to participating elements is reached. A value of "True" makes the algorithm activate twice the amount of elements (say $2 \cdot n$) in each step needed to achieve the target element ratio in the desired number of iteration steps. After recalculation the n lowest ranking elements get deactivated again for compensation.

5.18 Elimination of tension or compression members

The "Tension/Compression Eliminator"-component[9] removes elements from a model based on the sign of their axial force.



```
Convergency reached: true after 0 iterations
 Elements removed: 10 [-10/+0]
```

Figure 32: The "Tension/Compression Eliminator"-component.

These are the available input parameters:

[9]This component was devised and programmed by Robert Vierlinger. The following section is based on his written explanations.

"Iter" The removal of tensile or compressive elements works in an iterative fashion. The procedure stops either when no changes occur from one step to another or if the the maximum number of iterations "Iter" is reached.

"Ind" Indices of the elements that may be removed in the course of the procedure. By default the whole structure is included.

"LC" You can specify a special load case to consider. The default is "0" which means that the superposition of all loads is taken.

"Compr" If true, then only members under compression will be kept. Ohterwise only members under tension will survive. This value is false by default.

Like in ESO and BESO, elements selected for removal are assigned a negligible stiffness.

5.19 Eigen-modes of structures

The Eigen-modes of a structure describe the shapes to which it can be deformed most easily in ascending order. The first mode is the one which can be achieved most easily. The higher the mode number the more force has to be applied. Eigen-modes find application in two areas of the engineering discipline: First in structural dynamics, second in stability analysis which involves the determination of buckling loads [9].

[9]An Eigen-mode $\vec{x}$ is the solution to the matrix-equation $\underset{\sim}{C} \cdot \vec{x} = \lambda \cdot \vec{x}$ which is called the special eigen-value problem. Where $\underset{\sim}{C}$ is a matrix, $\vec{x}$ a vector and λ a scalar (that is a number) called eigen-value. The whole thing does not necessarily involve statical structures. Eigen-modes and eigen-values are intrinsic properties of a matrix. When applied to structures then $\underset{\sim}{C}$ stands for the stiffness-matrix whose number of rows and columns corresponds to the number of degrees of freedom of the statical system. $\vec{x}$ is an eigen-mode as can be computed with Karamba.

Vibration modes $\vec{x}$ of structures result from the solution of the generalized Eigenvalue problem. This has the form $\underset{\sim}{C} \cdot \vec{x} = \omega^2 \cdot \underset{\sim}{M} \cdot \vec{x}$. In a structural context $\underset{\sim}{M}$ is the mass-matrix which represents the effect of inertia. The scalar ω can be used to compute the eigen-frequency f of the dynamic system from the equation $f = \omega/2\pi$. In the context of structural dynamics eigen-modes are also called normal-modes or vibration-modes. If the eigenfrequency has a low value this means that the vibration takes a long time for one cycle to complete. In the limit when $f = 0$ this means that the static system leaves its initial state of equilibrium and never returns. In other words the system gets unstable. As the value of f goes towards zero so does the speed of vibration. This means that inertia forces do not play an important role. So if someone is interested whether a given statical system is stable or not can leave out the mass-matrix and solve the special eigenvalue problem instead of the general one (which is easier).

5.19.1 Eigen-modes in structural dynamics

If you hit a drum on its center the sound you hear originates from the drum-skin that vibrates predominantly according to its first normal-mode. When things vibrate then the vibration pattern and frequency depend on their stiffness and support conditions, the distribution of mass and where you hit them. Imagine a drum with a small weight placed on it: the weight will change the sound of the drum. The same is true when you hit the drum near its boundary instead of in the center: You will excite the drums eigen-modes differently.

5.19.2 Eigen-modes in stability analysis

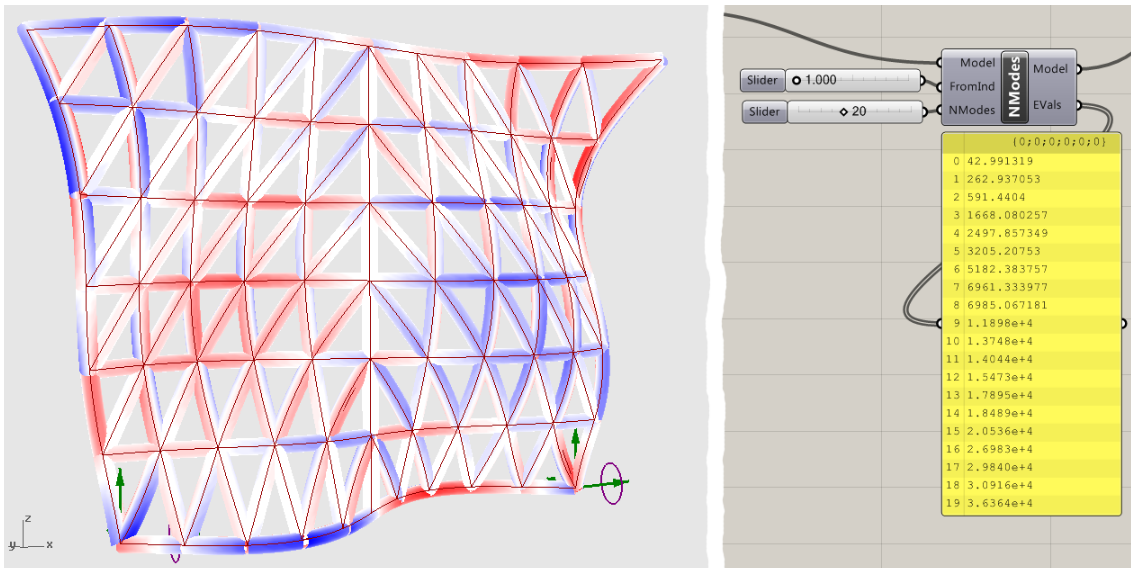
An unstable structure can be thought of as a statical system that vibrates very slowly, leaves its initial state of equilibrium and never comes back. This is why inertia forces do not play a role in the calculation of buckling-shapes. Otherwise the phenomina of vibration and buckling are very similar: The loads placed on a structure together with the eigen-modes determine the actual way a structure buckles.

The Euler-cases describe the buckling modes of beams subject to different support conditions. The more degrees of freedoms fixed at the boundary the higher the load at which a beam buckles.

5.19.3 The EigenMode-component

Karambas EigenMode-component allows to calculate eigen-modes and corresponding eigen-values of structures (see figure 33 and "NModesWall.ghx" in the examples section on the Karamba web-site) as used for buckling analysis i.e. without inertia effects. The input parameters are a model, the index of the first eigen-mode to be computed and the number of desired eigen-modes. The model which comes out on the right side lists the computed eigen-modes as load cases. Thus they can be superimposed using the ModelView-component for form-finding or structural optimization. All loads which were defined on the input model get discarded. The determination of eigen-shapes can take some while in case of large structures or many modes to be calculated.

The number of different eigen-modes in a structure equals the number of degrees of freedom. In case of beams there are six degrees of freedom per node, with only trusses attached a node possesses three degrees of freedom.



```
o 1.000
◇ 20
        {0;0;0;0;0;0}
 0 42.991319
 1 262.937053
 2 591.4404
 3 1668.080257
 4 2497.857349
 5 3205.20753
 6 5182.383757
 7 6961.333977
 8 6985.067181
 9 1.1898e+4
10 1.3748e+4
11 1.4044e+4
12 1.5473e+4
13 1.7895e+4
14 1.8489e+4
15 2.0536e+4
16 2.6983e+4
17 2.9840e+4
18 3.0916e+4
19 3.6364e+4
```

Figure 33: Left: 14th eigen-mode with strain display enabled. Right: EigenMode-component in action.

Figure 34 shows the first nine eigen-modes of a triangular beam mesh that is fixed at its lower corners. In the upper left corner of figure 34 one sees the undeflected shape. The higher the index of an eigen-mode the more folds it exhibits.

The eigen-values represent a measure for the resistance of a structure against being deformed to the corresponding eigen-form. Values of zero or nearly zero signal rigid body modes.

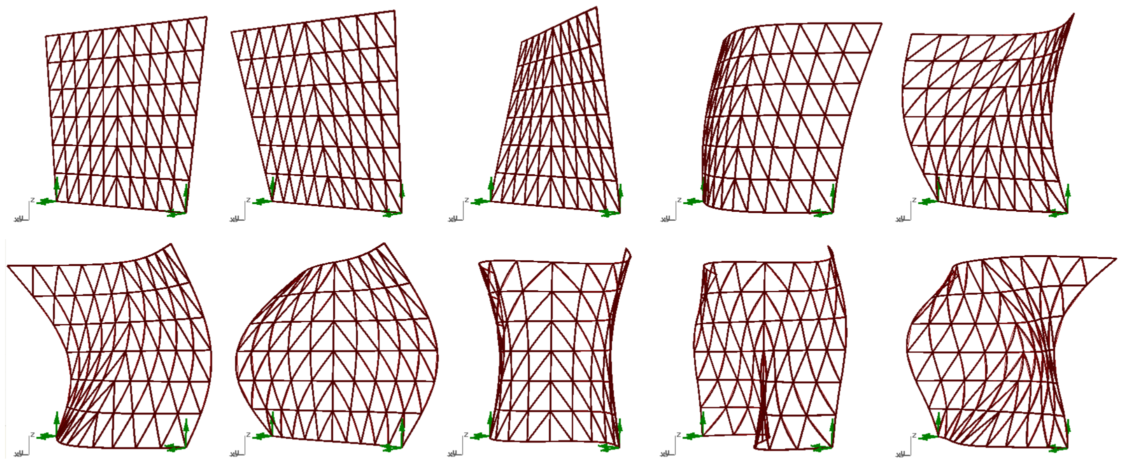


Figure 34: Undeflected geometry (upper left corner) and the first nine eigen-modes of the structure.

5.20 Hints on reducing computation time

Karamba spends most of its time solving for the deflections of a model. The time needed depends on the number of degrees of freedom n of the statical system and how many connections exist between the nodes. In the theoretical case that each node is connected to all others computation-time grows with n^3. If each node is connected to n_{neigh} others and the overall structure has a main axis along which it is oriented (i.e. there are no connections between distant nodes) then computational effort increases approximately with $0.5 \cdot n \cdot n_{neigh}^2$. Karamba makes use of multiple processors so having more than one saves time. Using trusses instead of beams more than halves computation time. When doing optimization with Galapagos the continuous display updates slow down things considerably. For better performance disable the preview of all components or minimize the Rhino window.

6 Utility Components

6.1 Nearest Neighbor

Assume you have a list of points and want to have a network that connects each point with a predefined number of its nearest neighbors or to points that lie within a given distance or both. In that case the "Nearest Neighbor"-component will be the right choice. It outputs lists of connection lines and a corresponding connectivity diagram. Be aware of the fact that these lists will probably contain duplicate lines. But this is no big problem (see below).

6.2 Remove duplicate lines

When you have a list of lines that you suspect of containing duplicate lines then send it through this component and out comes a purified list of ones of a kind. The input-plug "LDist" determines the limit distance for nodes to be considered as identical. Lines of length less than "LDist" will be discarded.

6.3 Remove duplicate points

Does essentially the same as the component described above – only with points instead of lines.

6.4 Line-line-intersection

This component takes a list of lines as input and mutually tests them for intersection. Output-plug "IP" delivers a list of intersection points. "LSS" returns all the line-segments including those which result from mutual intersection.

7 Trouble shooting

7.1 Karamba does not work for unknown reason

This is the recommended procedure:

1. If a component turns red read its runtime-error-message.
2. In case that more than one item is plugged into an input, check the incoming data via a panel component.
3. Sometimes flattening the input data helps: The dimension of input-lists must be consistent. For diagnosis plug them into a Panel-component which will show the dimensionality of the data. Another method is to enable "Draw Fancy Wires" in the View menu: Differently outlined connection lines signify different dimensionality of the data that flows through them.
4. If no results show, check whether preview is enabled on the ModelView-component.
5. If the Analyze-component reports a kinematic structure do the following:
   - Check the supports for forgotten support conditions.
   - Start to fix all degrees of freedom on your supports until the Analyze-component reacts.
   - Introduce additional supports.
   - Plug the model into the EigenModes-component. The first eigen-modes will be the rigid body modes you forgot to fix.
   - If the first few eigen-modes seemingly show an undeflected structure there might be beams in the system that can rotate about

their longitudinal axis. Enable "Local Axes" in the ModelView-component and move the slider for scaling Deformation in order to check this.

- Turn trusses into beams by activating their bending-stiffness. Be aware of the fact that a node has to be fixed by at least three trusses that do not lie in one plane.
- Remember that trusses have no torsional or bending stiffness and thus can not serve to fix the corresponding rotations on a beam that attaches to the same node.
- Check whether an element has zero area, height or Young's modulus.

7.2 "fem.karambaPINVOKE"-exception

On some computers the analysis component of Karamba refuses to work and throws a "fem.karambaPINVOKE" exception. This may be due to left-overs from previous Karamba installations which were not removed properly during the installation procedure. In such a case precede as follows:

- Uninstall Karamba completely via settings/Software/...
- Make sure that everything was removed:
  - Go to folder ".../Rhinoceros 4.0/Plug-ins/Grasshopper".
  - remove files "karamba.gha", "karamba.dll", "KDTreeDLL.dll" and "libiomp5md.dll" by hand if they still exist

This is plan b if the above does not help:

- Start Grasshopper
- Type 'GrasshopperDeveloperSettings' in the Rhino Window and hit 'ENTER'
- Toggle the status of the "Memoryload *.GHA assemblies using COFF byte arrays" option
- Restart Rhino

Plan c is to post a help request to the karamba group at `http://www.grasshopper3d.com/group/karamba?xg_source=activity`.

7.3 Karamba does not work after reinstalling Grasshoper

Upon installing Grasshopper some files of the Karamba package may get erased on some systems. Try to reinstall Karamba.

7.4 Predefined displacements take no effect

Check whether you disabled the correct degrees of freedom in the "Conditions" section of the PreDisp-component.

7.5 The ModelView-component consistently displays all load cases simultaneously

If the ModelView-component does not seem to react to selections done with the drop-down-list for load cases, check the value in the "LC-Index"-input plug. Remember that its value is added to the load case index selected on the drop-down-list. If the sum is negative all load cases will be displayed.

7.6 Icons in "Karamba"-toolbar do not show up

Sometimes it happens that Karambas component panels do not display any component icons. Select menu item "View/Show all components" in order to make them show up.

7.7 Error messages upon loading definitions saved with outdated Karamba versions

When loading definitions based on outdated Karamba version a pop-up window will inform you that "IO generated x messages,...". Normally this can be ignored. It may happen however that very old Karamba components do not load. In this case put their current versions in place.

7.8 Component in old definition reports a run-time error

On some components the order of input-plugs changed over time (e.g. the Assemble-component). They will turn red when loaded and the runtime error message will state that one object can no be cast to some other object. In this case replace the old component with a new one and reattach the input-plugs accordingly.

7.9 Other problems

In case you encounter any further problems please do not hesitate to contact us at `karamba@bollinger-grohmann-schneider.at`. or via the karamba group at `http://www.grasshopper3d.com/group/karamba?xg_source=activity`.

8 Version history

8.1 Version 0.9.06 - released on June 6, 2011

- Robert Vierlinger contributed a component for bidirectional evolutionary structural optimization.
- There are two new components that allow to define Materials:
  - the "Material"-component groups material properties.
  - the "MaterialSelect"-component lets you chose predefined materials from a data base in "csv"-format that can be easily extended via Excel or OpenOffice.

  A new ModifyBeam-component accepts them as input.
- The new subsection "results" contains two components for retrieving an elements normal force, resultant bending moment, internal bending energy and internal axial deformation energy. A third component outputs nodal displacements i.e. translations and rotations.
- Analysis has a new output-plug called "Energy" which returns the elastic energy stored in the structure. It is a measure for the overall stiffness: the larger its value the more flexible the structure. All output of the "Analysis"-component now comes for each load case separately.
- An error in the "NearestNeighbor"-component was removed.
- The subsection "ZZZ-deprecated" now holds all components that changed in previous versions and will be removed in future. They are kept for the time being so that definitions based on previous versions of Karamba remain valid.

8.2 Version 0.9.007 - released on September 7, 2011

- Beams can be given an identifier. This makes it sometimes easier to attach properties to them and set their visibility.

- The result of multiple "Line2Beam"-components can be plugged into an assembly.
- New options in the ModelView component:
  - Bending moments, shear and normal forces can be displayed along the beam axis. The corresponding output of the ModelView-component consists of meshes and curves and thus lends itself to further processing.
  - In addition to strains it is now also possible to visualize stresses and level of material utilization.
  - All color plots now also work with superimposed load cases.
  - The Mesh-output was reorganized: Instead of one big mesh a list of meshes with one entry per beam is now generated.
  - The input plug "Colors" can be used to customize the colors used for plotting stress, strain and utilization.
  - "Id" on the input side lets you select parts of the model for display. Regular expressions may be used to enable visibility for multiple groups of beams.
  - Two new output-plugs – "Legend C" and "Legend T" – can be used to show a legend for the color plots.
  - There is an option for displaying local beam axes. The new "Orientate Beam"-component gives the user control over the orientation of beams.
- Besides circular profiles there are now also box-, I- and trapezoid profiles available.
- The results output for displacements, cross section forces and internal energies was reorganized: There are now separate output-plugs for bending and translational components.
- A configuration file allows to customize the appearance (colors, number formats, etc.) of Karamba.
- The "Assemble"-component renders the mass of the structure in kilogram.
- When analyzing eigen-modes of a structures also eigen-values are available.